Thèse de doctorat

spécialité

Mathématiques, informatique.

sujet de thèse:

PROBLEMES D'OPTIMISATION COMBINATOIRES PROBABILISTES

Présentée par Monia Bellalouna

et soutenue le 5 Mars 1993

devant le jury composé de:

Nicolas Bouleau (Président)

Catherine Roucairol (Rapporteur)

Vangelis Paschos (Rapporteur)

Patrick Jaillet

Michel Minoux

Gérard Plateau

Remerciements

Je tiens tout particulièrement à remercier Patrick Jaillet, pour avoir dirigé ce travail.

J'ai le plaisir d'exprimer toute ma reconnaissance à Catherine Roucairol pour avoir accepter de rapporter ce travail, et qui a su me soutenir moralement aux moments les plus difficiles.

Vangelis Paschos a témoigné beaucoups d'intérêt pour mon travail. Je suis enchanté qu'il ait accepté d'être rapporteur et je le remercie chaleureusement.

C'est un très grand honneur pour moi que Nicolas Bouleau ait accepté d'être président de mon jury de thèse. J'aimerais exprimer tout mon respect et ma gratitude à Messieurs Michel Minoux et Gérard Plateau qui ont acceptés de participer à ce jury.

Je dois aussi remercier l'ensemble du CERMA, pour le cadre agréable et convivial de travail, que tous contribuent à offrir.

Enfin je remercie des êtres chers, qui se reconnaitront dans ces lignes, et savent l'importance qu'a leur amitié ou leur amour.

Table des Matières

Liste des sigles utilisés

- BF : Best-Fit.
- BFD : Best-Fit-Decreasing.
- FF :First-Fit.
- FFD :First-Fit-Decreasing.
- LPT : Largest-Processing-Time
- NF : Next-Fit.
- NFD : Next-Fit-Decreasing.
- PARPMP : Problème d'Arbre de Recouvrement de Poids Minimum Probabiliste.
- PBP : Problème du Bin-Packing.
- PBPB : Problème du Bin-Packing Probabiliste.
- POT : Problème d'Ordonnancement des Travaux.
- POTP : Problème d'Ordonnancement des Travaux Probabiliste.
- PTP : Le Problème de Tournée Probabiliste.
- PVC : Problème du Voyageur de Commerce.
- PVCP : Problème du Voyageur de Commerce Probabiliste.
- PPCCP : Problème du Plus Court Chemin Probabiliste.
- SPT : (*Shortest Processing Time*)
- WLPT : (*weighted Largest processing time*)
- WSPT : (*weighted Shortest processing time*)

Introduction

Depuis une quinzaine d'années l'étude des problèmes d'optimisation combinatoires n'a cessé de prendre de l'importance et est devenu l'une des branches des mathématiques discrètes les plus actives. Ce domaine d'activité s'est avéré prolifique tant dans l'introduction de nouveaux concepts que dans la résolution algorithmique d'un grand nombre de problèmes pratiques très complexes. Les résultats marquants de cette période comprennent : le développement de règles de base pour classifier et évaluer la difficulté intrinsèque des problèmes d'optimisation combinatoires (par exemple la hiérarchisation des classes de problèmes P, NP, NP-Complet, NP-Dur) ; l'approfondissement et l'éclaircissement de notions fondamentales telles que la complexité algorithmique et l'analyse de la performance des heuristiques ; l'introduction de techniques novatrices de résolution algorithmique qui sont devenues fondamentales dans bien d'autres domaines (par exemple la programmation dynamique, les méthodes de "branch and bound", le recuit simulé, etc.) ; et, enfin, l'utilisation de ces concepts et techniques de résolution dans des domaines d'applications très variés, souvent avec d'excellents résultats pratiques. La littérature scientifique du domaine s'est bien sûr développée.

Très récemment, des travaux se sont développés au sur une classe de problèmes d'optimisation combinatoires, caractérisée par le fait que des éléments probabilistes sont inclus explicitement dans leur formulation. Pour cette raison, ils ont été nommés 'Problèmes d'Optimisation Combinatoires Probabilistes' (POCPs).

Plusieurs motivations ont provoqué l'étude des effets de l'introduction d'éléments probabilistes dans des problèmes d'optimisation combinatoires. Deux d'entre elles sont particulièrement importantes et seront illustrées plus précisément tout au long de cette thèse. La première correspond au désir de formuler et d'analyser des modèles qui sont plus appropriés pour des problèmes pratiques pour lesquels l'aléatoire est une source constante de préoccupations. En fait, il y a une quantité importante d'applications intéressantes pour les POCPs, notamment dans divers contextes tels que la planification stratégique et tactique des services de collecte et/ou de desserte, les systèmes de communication et de transport, les systèmes de prévision de production d'ateliers, les réseaux distribués d'ordinateurs, etc. Pour de telles applications, des modèles de nature probabiliste sont plus particulièrement attractifs comme abstraction mathématique des systèmes réels. La seconde motivation est d'analyser la stabilité des solutions optimales des problèmes déterministes lorsque les exemplaires sont perturbés : les perturbations pour des problèmes définis sur des graphes sont simulées par la présence ou l'absence de sous ensembles de l'ensemble des sommets.

Description Générale de la Méthodologie

Très souvent dans les applications, après avoir résolu un exemplaire particulier d'un problème d'optimisation combinatoire donné, nous devons résoudre d'une façon répétée un grand nombre d'exemplaires du même problème. Ces autres exemplaires ne sont généralement que de simples variations du problème initial ; cependant ils sont suffisamment différents pour nécessiter un traitement individuel.

L'approche la plus naturelle pour répondre à ce genre de situation consiste à résoudre de façon optimale (ou pseudo-optimale à l'aide d'heuristiques) les différents exemplaires potentiels. Nous appellerons cette stratégie la "stratégie de ré-optimisation". On la notera par la lettre grecque Σ. Cette approche a néanmoins de nombreux désavantages. Par exemple si le problème de reconnaissance associé au problème considéré est $NP-$complet, on peut avoir à résoudre un nombre exponentiel d'exemplaires d'un problème réputé très difficile. De plus dans beaucoup d'applications il est nécessaire d'obtenir une solution à chaque exemplaire d'une façon très rapide, et souvent le matériel informatique n'est pas disponible (ou n'existe pas : certain problèmes $NP-$complets de grande taille sont tout simplement hors de portée du domaine "traitable" de l'informatique) pour réaliser une telle tâche.

Il est donc nécessaire d'adopter une stratégie différente. Plutôt que de réoptimiser chaque exemplaire successif, on peut essayer de déterminer une solution *a priori* du problème initial qui puisse se modifier successivement d'une façon simple pour résoudre les exemplaires suivants. Nous appellerons cette stratégie une "stratégie a priori".

Naturellement les questions qu'on est en droit de se poser sont les suivantes : quelles mesures doit on choisir pour évaluer une telle stratégie a priori ? **comment modifier la solution a priori** pour répondre aux problèmes suivants ?

Pour une illustration de la méthodologie, on va se restreindre à une classe de problèmes définis sur des graphes valués. Un graphe $G = (V, E)$ de n sommets dont les arcs sont valués et sur lequel est défini un problème d'optimisation (par exemple le problème du voyageur de commerce, qu'on notera PVC). Si seul un sous-ensemble de sommets de V est présent lors de la résolution d'un exemplaire donné du problème (par exemple, un jour donné, le voyageur peut n'avoir à visiter qu'un sous-ensemble S des sommets V), alors il peut y avoir 2^n exemplaires différents possibles (chacun correspondant au sous-graphe de G défini par les 2^n sous-ensembles possibles de V). Supposons que chaque exemplaire S a une probabilité d'occurrence $\mathbb{P}(S)$. Supposons aussi que nous avons une méthode $\mathcal{U}$ pour modifier une solution réalisable f du problème défini sur l'ensemble V de tous les sommets en une solution réalisable f_S du problème défini sur un sous-ensemble quelconque S de l'ensemble des sommets. Si on note $L_f(S)$ la valeur (le "coût") de f_S (dans le cas du PVC, f_S est une tournée des sommets de S et $L_f(S)$ sa longueur), le choix naturel pour sélectionner f parmi toutes les solutions réalisables possibles est de minimiser le coût moyen défini par :

$$\mathbb{E}[L_f] = \sum_{S \subseteq V} \mathbb{P}(S) L_f(S). \tag{0.1}$$

En d'autres termes on aimerait minimiser la somme pondérée des valeurs $L_f(S)$, obtenues en appliquant la méthode de modification $\mathcal{U}$ à une solution réalisable f.

Le choix de (0.1) donne une réponse raisonnable à la question d'une mesure d'évaluation de notre stratégie. Mais quelles propriétés devrait avoir la méthode $\mathcal{U}$? Une qualité idéale serait d'assurer que $L_f(S)$ soit "proche" de la valeur de la solution optimale $L_{Opt}(S)$ du

problème ré-optimisé pour tout exemplaire S. Une qualité moins restrictive et plus globale serait d'avoir $\mathbb{E}[L_f]$ "proche" du coût moyen obtenu sous la stratégie de ré-optimisation et défini par :

$$\mathbb{E}[\Sigma_{Opt}] = \sum_{S \subseteq V} \mathbb{P}(S) L_{Opt}(S). \tag{0.2}$$

De plus, $\mathcal{U}$ doit pouvoir modifier f d'une façon efficace d'un exemplaire à un autre. Le choix des $\mathcal{U}$ dans les problèmes que nous allons décrire plus loin peut sembler arbitraire. En fait ces choix sont tout à fait naturels. Ils permettent de faire des modifications très simples et correspondent à des pratiques courantes dans les applications. Leurs qualités vis à vis des stratégies de ré-optimisation sont de plus tout à fait satisfaisantes.

Présentation de la Thèse

Les aspects novateurs et stimulants des POCPs nous ont poussés à nous y intéresser. Nous allons continuer et approfondir le travail sur les POCPs dans la direction des résultats obtenus jusqu'à présent, mais aussi nous allons explorer des domaines entièrement nouveaux.

Depuis l'introduction de cette méthodologie due à Patrick Jaillet et Amedeo Odoni 1985, les travaux sur le sujet ont été considérable. Ainsi nous consacrons le premier chapitre de cette thèse à une présentation des problèmes étudiés et nous indiquerons les différentes questions abordées pour ces problèmes.

Le deuxième chapitre constitue une contribution au Problème du Voyageur de Commerce Probabiliste (PVCP). Nous exhibons des bornes pour la fonctionnelle associée au PVCP, qui nous donne une comparaison de la solution du PVCP et de la solution du PVC. Le PVCP étant $NP-complet$, il est intéressant de savoir s'il existe des sous classes de ces problèmes qui appartiennent à P, ceci permet de délimiter le domaine "traitable" du problème. Notre but est de chercher une structure particulière et des valuations spécifiques qui rendent le problème facile. Nous étudions des PVCs faciles et nous montrons que pour ces cas particuliers, les deux stratégies de réoptimisation et a priori coïncident. Nous pouvons ainsi déduire que la méthode de modification choisie pour le PVCP n'est pas arbitraire.

En général, et pour une loi de probabilité quelconque, on ne connaît pas d'heuristique en pire cas constant pour le PVCP. Nous exhibons une classe de probabilité où l'algorithme de Christophide est un algorithme en pire cas constant.

Nous nous sommes intéressés à l'implémentation d'heuristiques pour le PVCP. Nous étudions, en particulier, l'algorithme du recuit simulé qui est très performant pour le PVC. La première partie de notre étude est consacrée à la recherche des paramètres optimaux de cette heuristique : nous proposons sur la base de nombreuses expérimentations un programme de refroidissement efficace. La fonctionnelle (la moyenne de la longueur d'un tour à travers n sommets) au sens PVCP étant fonction de la probabilité p de présence de chaque ville et de $n-1$ quantités liées au tour, le calcul de la différence de coût de deux solutions voisines est au meilleur des cas de l'ordre de n (pour le PVC cette différence se calcule en un temps constant indépendant de n). Nous mettons en oeuvre et comparons plusieurs structures de voisinage,

deux-inter-change, shift et structure de cycle, mais dans tous les cas le recuit simulé reste peu performant pour le PVCP à cause d'une grande complexité de calcul. Des approximations au premier (ce qui revient à approximer le PVCP par le PVC) ou second ordre, permettent de réduire la complexité des calculs sans altérer dans la pratique la qualité de la solution. Les résultats expérimentaux laissent penser que le tour optimal du PVC euclidien est une assez bonne heuristique pour le PVCP euclidien même pour des probabilités p petites.

Nous implémentons ensuite l'algorithme tabou. Cette heuristique s'est avérée particulièrement efficace dans les nombreuses expérimentations effectuées et les performances surpassent nettement celles de l'algorithme du recuit simulé. Nous obtenons la même performance quand nous nous limitons à une approximation de la fonctionnelle pour le recuit simulé.

Ces deux algorithmes deviennent impraticables pour des problèmes de taille supérieure à 200, ce qui est généralement le cas dans les applications réelles. Nous implémentons l'algorithme de partitionnement de Karp, qui est particulièrement adapté pour les problèmes de très grandes tailles. Cet algorithme qui n'est pas très performant du point de vue qualité de la solution, converge très rapidement.

Dans le troisième chapitre, nous nous intéressons à définir des stratégies a priori qui nous permettent d'obtenir des problèmes qui par définition donnent des solutions optimales stables.

Il est démontré que les problèmes de reconnaissance associés à tous les POCPs déjà étudiés sont en toutes généralités $NP-$complets. Ces résultats sont particulièrement surprenants pour le problème du plus court chemin probabiliste (PPCCP) et le problème d'arbre de recouvrement de poids minimal probabiliste (PARPMP) pour lesquels la version déterministe est dans P et est très facilement résoluble. On a aussi montré, pour ces problèmes, que le calcul de l'expression (0.2) est un problème d'énumération très difficile. Nous avons ainsi pour but de chercher des problèmes tels que l'introduction de probabilités dans leur formulation n'en change pas la complexité. Les résultats que nous présentons dans le chapitre 3 répondent à cette caractéristique.

Nous étudions une stratégie a priori pour le Problème d'Ordonnancement des Travaux (POT) à une machine, que nous noterons **Problème d'Ordonnancement des Travaux Probabiliste** (POTP). Nous obtenons une expression explicite de la fonctionnelle associée au POTP à une machine et nous démontrons que le POT à une seule machine est un problème stable : la solution optimale du problème déterministe est la solution du problème probabiliste, pour une loi de probabilité quelconque. Nous étudions ensuite le POTP à plusieurs machines qui se réduit à un POT à plusieurs machines pondéré et il est donc $NP-$complet. Nous en déduisons que, dans le cas où toutes les tâches ont la même probabilité de présence, le problème se réduit à un POT à plusieurs machines pondéré de même pondération et nous obtenons donc un problème facile. En plus, dans ce cas la solution optimale du problème déterministe est la solution de notre stratégie a priori.

Nous avons aussi obtenu une expression analytique explicite pour la fonctionnelle associée à la stratégie de réoptimisation du POT à plusieurs machines. Ceci nous permet d'avoir une comparaison explicite des deux stratégies a priori et de réoptimisation. Nous montrons que le calcul de la fonctionnelle de la stratégie de réoptimisation du POT se fait en temps poly-

nomial, dans le cas de machines identiques ou uniformes. Nous obtenons ainsi le premier POCP dont le calcul de l'équation (0.2) est dans P. Ainsi cette étude permet d'avoir une caractérisation des problèmes stables : tous les problèmes qui sont résolubles par un algorithme de liste imperturbable par l'absence de certaines données sont des problèmes stables. Cette caractérisation nous a permis ensuite de déduire d'autres problèmes stables.

Dans le chapitre IV nous nous intéressons à la généralisation de la méthodologie a priori à des POCs non définies sur des graphes. Le problème que nous allons considérer est **Le Problème du Bin-Packing Probabiliste** (PBPP). Nous allons commencer par décrire les motivations de ce choix.

L'analyse asymptotique des POCs est devenue depuis quelques années un domaine de recherche important et fructueux. Les résultats obtenus dans ce domaine sont très utiles pour plusieurs raisons, ils permettent d'obtenir des approximations pour des problèmes de très grande taille, d'analyser les performances de certaines heuristiques et enfin d'explorer la frontière entre bons et mauvais algorithmes au sens probabiliste. On remarque que le Problème de Bin Packing (PBP) joue un rôle très important dans ce domaine : on sait que pour le problème de Bin Packing et sous certaines conditions pour plusieurs heuristiques, le nombre de lots obtenue par une heuristique H divisé par n (n est le nombre d'objet) tend presque sûrement vers une constante. On note que la constante est explicite pour plusieurs heuristiques .

C'est pourquoi nous nous intéressons à une approche a priori pour ce problème. Nous décrivons plusieurs stratégies de redistribution basées sur des heuristiques très performantes du PBP. Nous analysons ces stratégies de redistribution : calcul de la fonctionnelle associée et étude de leurs complexités. Nous nous sommes intéressé particulièrement à l'analyse asymptotique et probabiliste de ces différentes stratégies. Sous la condition que les objets sont uniformément et indépendamment répartis dans $[0,1]$ nous montrons que la fonctionnelle associée à la stratégie de redistribution suivant Next-Fit-Decreasing tend presque sûrement vers une constante. On note que cette constante est donnée explicitement en fonction de la constante de la fonctionnelle quand tous les objets sont présents. Une évaluation précise de la déviation de la fonctionnelle de sa moyenne est donnée. En plus nous démontrons que la fonctionnelle centrée converge en loi.

Nous étudions des stratégies asymptotiquement équivalentes à la stratégie de ré-optimisation, ceci nous a permis d' effectuer une analyse asymptotique explicite de la stratégie de ré-optimisation. Cette étude nous permet de mieux comprendre le comportement asymptotique des deux stratégies de redistribution et de ré-optimisation, et confirme nos intuitions sur ces comportements : asymptotiquement la solution d'une heuristique H à travers la moyenne des données est une très bonne estimation de la stratégie de redistribution suivant H.

Nous finissons par souligner que cette façon nouvelle et innovatrice d'introduire les probabilités dans la formulation des POCs est loin d'avoir été complétement explorée, nous le détaillons dans la conclusion après un bref résumé de nos recherches.

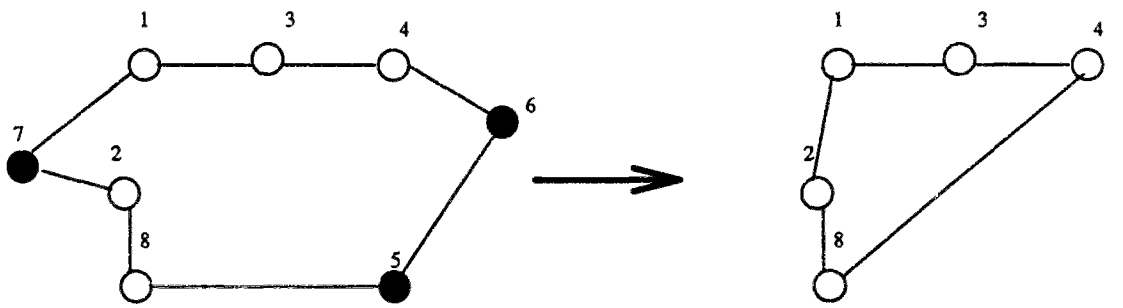
Chapitre 1

Rappel de la Méthodologie a priori

Dans ce chapitre nous allons rappeler brièvement quelques problèmes (parmi d'autres) pour lesquels la méthodologie a priori a été appliqué avec succès ([39],[9],[7],[5]). Ensuite on donnera les différents questions abordés pour ces problèmes. Les problèmes génériques sont les suivants :

Le Problème du Voyageur de Commerce Probabiliste

Soit le problème de visiter n sommets donnés d'un graphe complet valué. Par la suite, seuls certains sous-ensembles $S_1, S_2, \ldots$, des n sommets nécessiteront réellement une visite, ces ensembles étant choisis successivement et indépendamment selon une même probabilité définie sur 2^V, l'ensemble de tous les sous-ensembles de V. Nous voulons obtenir, avant de connaître la suite $(S_i)_i$, une tournée parmi les n sommets initiaux telle que, si S est le sous-ensemble à visiter, ses sommets soient parcourus dans le même ordre que celui défini par la tournée a priori. Le problème de trouver une tournée a priori minimisant l'espérance des distances parcourues sous cette stratégie est appelé le problème du voyageur de commerce probabiliste (PVCP) [34]. La méthode de modification $\mathcal{U}$ consiste donc simplement à "gommer" de la tournée initiale les sommets qui n'appartiennent pas à S.



Un tour a-priori

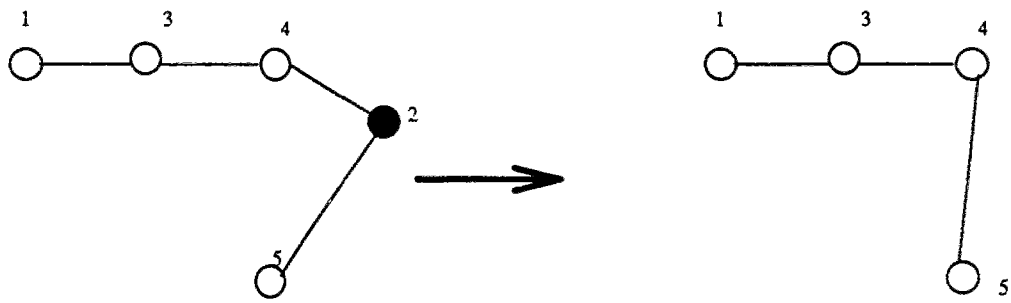
le tour resultant quand 5,6,7 sont absents

Nous aurons l'occasion de revoir plus en détail ce problème dans le chapitre suivant.

Le Problème du Plus Court Chemin Probabiliste

Soit le problème de choisir un chemin entre un sommet initial s et un sommet final t dans un graphe complet valué, ayant n sommets supplémentaires ($G = (N, E)$, avec $N = V \cup \{s, t\}$). Par la suite, seuls certains sous-ensembles $S_1, S_2, \ldots$, de ces n sommets supplémentaires pourront vraiment être utilisés dans un chemin entre s et t (les autres sommets étant "indisponibles"), ces ensembles étant choisis successivement et indépendamment selon une même probabilité définie sur 2^V.

Nous voulons obtenir, avant de connaître la suite $(S_i)_i$, un chemin entre s et t tel que, si S est le sous-ensemble des sommets "empruntables", on élimine du chemin initial tous les sommets de $V \setminus S$ tout en conservant l'ordre de succession des autres sommets du chemin. Le problème de trouver un chemin a priori entre s et t minimisant l'espérance des distances parcourues sous cette stratégie est appelé le problème du plus court chemin probabiliste (PPCCP). La méthode de modification $\mathcal{U}$ consiste donc simplement à "gommer" du chemin initial les sommets qui n'appartiennent pas à S.



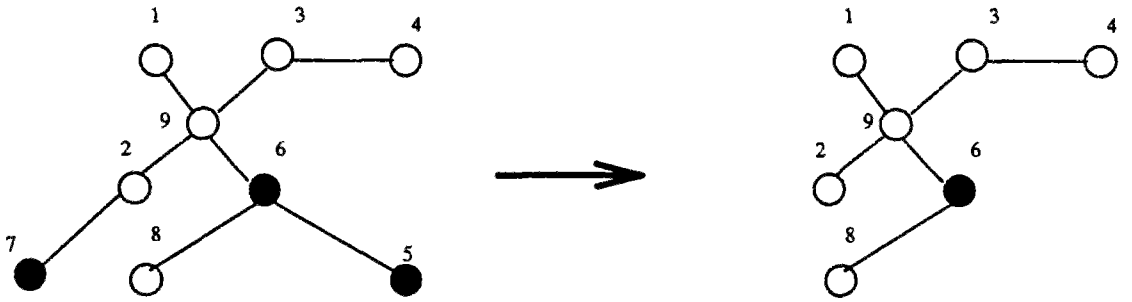
Un chemin a-priori

le chemin resultant quand 2 est absent

Le Problème d'Arbre de Recouvrement de Poids Minimum Probabiliste

Ce problème est une généralisation naturelle du problème classique d'arbre de recouvrement de poids minimal. Etant donnés n sommets d'un graphe non-orienté valué $G = (V, E)$, le problème consiste à trouver un arbre reliant tous les sommets de G. Cependant par la suite, seuls certains sous-ensembles étant choisis successivement et indépendamment selon une même probabilité définie sur 2^V. Le problème consiste à construire un arbre de recouvrement de G avant de connaître la suite $(S_i)_i$ et de l'utiliser de la façon suivante : si S est l'ensemble des sommets devant être reliés on élimine de l'arbre initial tous les sommets de $V \setminus S$ de telle façon que l'arbre reste connecté. Le problème de trouver un arbre a priori minimisant l'espérance des longueurs sous cette stratégie est appelé problème d'arbre de recouvrement de poids minimal probabiliste (PARPMP) (cf [6]). La méthode de modification consiste simplement à "gommer" de l'arbre initial tous les sommets qui n'appartiennent pas à S tout en assurant que l'arbre reste connecté.

D'autres problèmes on était abordés, on peut citer le **Le Problème de Tournée Probabiliste** [36], et **Le Problème de la Localisation du Voyageur de Commerce Probabiliste** [5].



Un arbre a-priori

l'arbre resultant quand 5,6,7 sont absents

Différents Résultats

On va présenter les différents questions abordées pour ces problèmes.

1. **Complexité des différentes stratégies**

   Il est démontré que les problèmes de reconnaissance associés sont en toutes généralités $NP-$complets. Ces résultats sont particulièrement surprenants pour le PPCCP pour lesquels la version déterministe est dans P et est très facilement résoluble. De tous ces problèmes c'est néanmoins le PPCCP qui semble être le plus "facile" dans le sens où on a trouvé une classe de probabilités pour lesquelles ce problème devenait "traitable" (i.e. dans P). On peut aussi démontrer que le calcul du coût de la stratégie de réoptimisation (0.2) est $\#P-$complet, c'est à dire, est un problème d'énumération très difficile.

2. **Analyse asymptotique et probabiliste des différentes stratégies**

   Ces résultats traitent des POCPs définis dans un espace euclidien [35]. Les n sommets sont des points de l'espace et les longueurs des arêtes sont données par la distance euclidienne. On suppose que chaque point a une probabilité p d'être présent. De plus on suppose que les positions des points sont uniformément et indépendamment distribuées sur le carré unité. Il est possible alors de démontrer que les fonctionnelles (données par l'équation (0.1)) des solutions optimales des POCPs divisées par $\sqrt{n}$ convergent toutes presque sûrement vers des constantes. Ces résultats ont été généralisés lorsque les points étaient uniformément répartis dans un ensemble Lebesgue-mesurable quelconque d'un espace euclidien de dimension $d \geq 2$, les constantes ne dépendant que de la dimension de l'espace et de la probabilité de présence des points, mais pas de la forme de l'ensemble. Des résultats similaires ont été obtenus pour les fonctionnelles associées aux stratégies de réoptimisation, mais la comparaison asymptotique des deux stratégies reste encore mal comprise.

3. **Propriétés combinatoires des POCPs**

   Un certain nombre de propriétés combinatoires importantes des POCPs ont été établis. Par exemple : des bornes précises sur les fonctionnelles des POCPs ; une caractérisation des POCPs dans les deux cas limites où les "probabilités de présence" sont très petites ou très grandes.

4. **Construction et analyse d'algorithmes**

   Il a été proposé pour tous les POCPs soit des algorithmes de résolution exactes basés sur la programmation dynamique ou sur des méthodes "branch and bound", soit des heuristiques pour lesquelles il y a des analyses de performance en pire cas.

5. **Implémentation des algorithmes**

   De nombreux algorithmes pour le PVCP ont été programmés et ont obtenus d'excellents résultats sur des problèmes euclidiens de plusieurs centaines de sommets. En particulier une heuristique de recherche locale donne des solutions entre 2% et 5% de l'optimalité.

Dans la suite on s'intéressera à l'étude de quelques points encore mal compris dans cette méthodologie. Nous programmons plusieurs algorithmes pour le PVCP euclidien où une implémention optimale est obtenue. Nous nous concentrons sur l'analyse de sous classes "traitables" (basées sur des probabilités particulières ou sur des structure particulières) des POCPs, puis nous tenterons d'établir une caractérisation de la stabilité des solutions de problèmes d'optimisation combinatoires. Enfin nous noous intéressons à une analyse asymptotique explicite : où nous démontrons que les constantes vers lesquelles tendent les fonctionnelles du PBPP sont données explicitement.

Chapitre 2

Problème du Voyageur de Commerce Probabiliste

Le Problème du Voyageur de Commerce (PVC), consiste à chercher le tour de longueur minimale à travers un nombre n fixe de villes. Plusieurs nouvelles idées en optimisation combinatoire ont été testées sur ce problème telles que la programmation dynamique [31], l'analyse asymptotique [3], le recuit simulé... Une extension probabiliste très naturelle du PVC est introduite pour la première fois par Jaillet [34], où il suppose que le nombre de villes présentes est une variable aléatoire.

Dans ce chapitre, on s'intéresse à cette extension probabiliste, intitulée Problème du Voyageur de Commerce Probabiliste (PVCP). Nous commençons par rappeler les résultats importants de Jaillet [34],[35],[41]. Dans le paragraphe 2.2 nous examinons certaines propriétés combinatoires en donnant des bornes du PVCP, afin de mieux comprendre la relation entre le PVCP et le PVC. Dans le paragraphe 2.3 nous examinons certains cas solvables. Nous remarquons que pour certains PVCs faciles, les PVCPs associés restent faciles. Nous montrons que dans ces cas, la stratégie a priori et la stratégie de ré-optimisation sont équivalentes et donc que ces problèmes sont stables. On donne dans [40] une classe de probabilités pour laquelle le PPCCP devenait "traitable". Cette classe de probabilités a la particularité de permettre une transformation polynomiale du PPCCP initial en une version déterministe équivalente de taille bornée par une fonction polynomiale de la taille initiale. Nous montrons que cette classe de probabilité nous permet de vérifier que l'heuristique de Christophides [49] est une heuristique en pire cas constant pour le PVCP. Finalement dans le paragraphe 2.4 nous analysons et implémentons, pour le PVCP euclidien, plusieurs heuristiques du type recuit simulé, algorithme tabou, algorithme de partitionnement de Karp.... Le recuit simulé et l'algorithme tabou sont des techniques générales d'optimisation combinatoire. Le recuit simulé est une heuristique qui fournit de bonne approximations pour le PVC en un temps raisonnable, mais cette propriété n'est plus vérifiée pour notre problème. En effet la performance de l'algorithme du recuit est basée sur un bon choix du voisinage, or pour le PVCP les solutions voisines sont fortement liées. Le voisinage de deux-inter-change nous donne un temps de calcul très important, nous proposons alors un autre voisinage qui améliore la complexité et ne change presque pas la qualité de la solution. Nous adaptons aussi la technique Tabou au PVCP pour ces deux voisinages. Nous implémentons également l'algorithme de partitionnement de Karp [46], cet algorithme est asymptotiquement optimal pour le PVC (pour le PVCP ce résultat n'est pas encore démontré). Nous présentons les résultats expérimentaux

dans le paragraphe 2.4 et nous discutons certaines propriétés qualitatives du PVCP tirées des résultats de ces heuristiques.

2.1 Rappel

Nous donnons ici quelques définitions et théorèmes importants concernant le PVCP. Ce préambule constituera la base de tout ce chapitre.

Définition du PVCP

On considère un graphe $G = (V, E)$ de n sommets dont les arcs sont valués, ce qui revient à se donner une matrice D de distances entre les sommets de G. On se donne une loi de probabilité $\mathbb{P}$ sur 2^V, l'ensemble de tous les sous-ensembles de V, i.e chaque exemplaire $S \subseteq V$ a une probabilité d'occurrence $\mathbb{P}(S)$. Etant donné un tour a priori T à travers les n sommets et pour chaque exemplaire S du problème, la méthode de modification $\mathcal{U}$ pour générer une solution à travers S consiste à "gommer" les villes absentes du tour a priori T.

Soit L_T la variable aléatoire réelle définie sur 2^V, qui à un tour a priori T et pour tout S de 2^V, associe la longueur du tour à travers S, $L_T(S)$, induit du tour T par la méthode de modification $\mathcal{U}$. Le problème du voyageur de commerce probabiliste consiste à chercher un tour a priori T qui minimise l'espérance de L_T.

Problème PVCP : Soient un graphe $G = (V, E)$, une loi de probabilité $\mathbb{P}$ sur 2^V et une méthode de modification $\mathcal{U}$. Le PVCP consiste à déterminer un tour a priori qui minimise l'espérance de la longueur du tour :

$$\min_T(\mathbb{E}(L_T) = \sum_{S \subset \{1,\ldots,n\}} \mathbb{P}(S) L_T(S))$$

Motivations

Pourquoi le choix de cette méthodologie ? Il est clair que l'approche la plus naturelle est la "stratégie de ré-optimisation" qui consiste à résoudre de façon optimale (ou pseudo-optimale à l'aide d'heuristiques) les différents exemplaires potentiels. Mais le problème de reconnaissance associé au problème du voyageur de commerce est $NP-$complet et on peut avoir à résoudre un nombre exponentiel d'exemplaires d'un problème réputé très difficile. En plus la "stratégie a priori" modélise beaucoup de problèmes concrets. Par exemple, la stratégie utilisée par un facteur n'est autre que la stratégie PVCP. En effet un facteur a un nombre fixe d'adresses sous sa responsabilité et chaque jour seul un sous ensemble de ces adresses est à visiter, il suit alors son tour habituel en sautant les adresses qui n'ont pas de courrier. Il existe beaucoup d'autres applications pour le PVCP [34].

Espérance d'un tour

On se donne un tour T, la variable aléatoire longueur du tour L_T peut prendre 2^n valeurs différentes. Pour chaque exemplaire S on a une longueur $L_T(S)$ qui se calcul en $0(n)$ additions. Donc le calcul de $\mathbb{E}(L_T)$ par énumération complète nécessiterait $0(n2^n)$ opérations. Une méthode plus efficace est proposée par Jaillet [34] qui réduit le calcul à $0(n^2)$.

Théorème 1 *(Jaillet [34]) On considère n villes et soit $D = (d_{ij})$ la matrice distance à travers ces n villes. Soit p_i la probabilité de présence de la ville i. Sans perte de généralité on suppose que $T = (1, 2..., n, 1)$. L'espérance de la longueur du tour T est donnée par la formule suivante :*

$$\mathbb{E}(L_T) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) + \sum_{j=1}^{n} \sum_{i=1}^{j-1} d_{ji} p_i p_j \prod_{k=j+1}^{n} (1 - p_k) \prod_{k=1}^{i-1} (1 - p_k) \qquad (2.1)$$

L'idée de la démonstration est que la distance d_{ij} contribue dans la fonctionnelle $\mathbb{E}(L_T)$ si et seulement si i est présent (probabilité p_i), j est présent (probabilité p_j) et les villes entre i et j $(i + 1, ...j - 1)$ sont absentes (probabilité $\prod_{k=i+1}^{j-1} (1 - p_k)$).

On se restreindra dans la suite de ce chapitre au cas où toutes les villes ont la même probabilité de présence ($p_i = p \ \forall \ i$), on notera $q = 1 - p$. Dans ce cas on a :

$$\mathbb{E}(L_T) = p^2 \sum_{r=0}^{n-2} q^r L_{T^r} \qquad (2.2)$$

avec

$$L_{T^r} = \sum_{i=1}^{n} d(i, T^r(i)) \qquad (2.3)$$

T^r consiste à sauter r villes du tour initial T, donc T^r est formé de $pgcd(n, r + 1)$ sous-tours. $T^r(i)$ la ville après i le long de la permutation T^r, donc L_{T^r} est la longueur de la permutation T^r. On remarque que T^0 est le tour T et L_{T^0} est la longueur du tour T.

Propriétés Combinatoires des PVCPs

Il est intéressant de comprendre le lien entre le PVC et le PVCP. Il est clair que le PVC est un cas particulier du PVCP quand tous les sommets sont présents. On a trouvé des cas très particuliers où le PVCP et le PVC sont identiques.

Proposition 1 *(Jaillet [34]) On considère le problème avec m villes toujours présentes, et n villes présentes avec une probabilité p, le PVC résout le PVCP pour tout graphe $G = (V, D)$ $(PVC \equiv PVCP)$ si et seulement si :*

(i) D symétrique et $n + m \leq 4$; si $m = 0$ ceci reste vrai pour $n = 5$.

(ii) D non symétrique et $m + n \leq 3$.

Proposition 2 *(Jaillet [34])*
Si les villes sont placées sur leur enveloppe convexe alors $PVC \equiv PVCP$.

Dans [34] on exhibe des exemples où le PVC est une très mauvaise solution pour le PVCP. La proposition suivante permet d'évaluer l'erreur effectuée quand on utilise le tour PVC comme solution du PVCP. On notera dans toute la suite de ce chapitre T_{PVCP} le tour optimal au sens PVCP et T_{PVC} le tour optimal au sens PVC.

Proposition 3 *(Jaillet [34]) Si D satisfait l'inégalité triangulaire, soit m le nombre de sommets toujours présents, alors pour $m \geq 1$ ou ($m = 0$ et n premier) on a :*

$$I\!E(L_{T_{PVCP}}) \geq p(1-(1-p)^{n-1})L_{T^0_{PVC}} \tag{2.4}$$

$$\frac{I\!E(L_{T_{PVC}}) - I\!E(L_{T_{PVCP}})}{I\!E(L_{T_{PVCP}})} \leq \frac{1-p}{p}$$

$$\frac{L_{T^0_{PVCP}} - L_{T^0_{PVC}}}{L_{T_{PVC}}} \leq \frac{1-p}{p^2}$$

Ces bornes sont les meilleures possibles (voir [34]). On remarque que si p est proche de 1, l'erreur $\frac{1-p}{p}$ est proche de zéro et donc le tour PVC est une très bonne solution approchée pour le PVCP. Si p est très petit l'erreur, $\frac{1-p}{p}$ tend vers l'infini et on ne dispose plus d'information sur le comportement du tour optimal au sens PVC comme solution de PVCP.

Le PVCP paraît beaucoup plus complexe que le PVC, les résultats suivants caractérisent la différence énorme entre les deux problèmes.

Proposition 4 *(Jaillet [34])*

Si la distance est euclidienne le tour optimal au sens PVCP peut s'intersecter (les arrêtes du tour optimal peuvent se couper).

Proposition 5 *(Jaillet [34])*

Soit $h = (1,...n)$ un chemin à travers n sommets, si on décompose h en deux chemins $h_1 = (1,..k)$ et $h_2 = (k,...,n)$ et si on note $h = h_1 \oplus h_2$ alors :

$$I\!E(L_{h_1 \oplus h_2}) \geq I\!E(L_{h_1}) + I\!E(L_{h_2}) \tag{2.5}$$

La fonctionnelle associée au PVCP n'est pas additive et on ne peut décomposer le PVCP en sous problèmes. Le principe d'optimalité n'est donc pas vérifié :

Proposition 6 *(Jaillet [34]) L'approche de programmation dynamique proposée pour le PVC est impossible pour le PVCP.*

Analyse asymptotique et probabiliste des différentes stratégies

Le but de ce travail était de trouver des résultats de convergence pour le PVCP, analogues à ceux démontrés par Beardwood et al [3] pour le PVC qui ont été d'un grand apport pour évaluer les performances de certaines heuristiques.

On rappelle le résultat de Beardwood et al [3] : si les positions des sommets sont des variables aléatoires indépendantes et uniformément distribuées sur le carré unité et si les longueurs des arêtes sont données par les distances euclidiennes, alors la longueur du tour optimal à travers n sommets (au sens PVC) normalisée par $\sqrt{n}$ tend presque sûrement vers une constante β. Cette constante β est estimée à 0.71. Ces travaux ont été ensuite généralisés par Steele [60] pour d'autres problèmes d'optimisation combinatoires.

Afin d'exposer les résultats similaires pour le PVCP démontrés par Jaillet [35],[41] on introduit les notations suivantes :

- IR^2 l'espace euclidien de dimension deux.
- $[0,r]^2$ le carré de coté r.
- $x = \{x_1, x_2, ...\}$ une suite de points de IR^2 et $x^n = \{x_1, ..., x_n\}$. Si la position des points est aléatoire la suite est représentée par $X = \{X_1, X_2, ...\}$.
- Pour une suite x, $T(x^n)$ représente un tour à travers les points de la suite x^n ; $\mathbb{E}(L_T(x^n, p))$ est l'espérance de la longueur du tour $T(x^n)$ où chaque point est présent avec une probabilité p indépendamment des autres points.
- On pourra introduire l'espace de probabilités $(\Omega, \mathcal{A}, \mathbf{P})$ où $\Omega := [0,r]^2$ et $\mathcal{A}$ est sa tribu Borélienne sur lequel les X_i sont des v.a et on définira les suites $X^{(n)}$ sur l'espace induit $(\Omega^{I\!N}, \mathcal{A}^{I\!N}, \mathbf{P}^{I\!N})$. Pour éviter toute confusion on notera $\mathbf{Eu}(.)$ l'espérance relative à la mesure de probabilités $\mathbf{P}$.

Proposition 7 *(Jaillet [34])*

$$\begin{aligned} I\!E(L_{T_{PVCP}}(x^n, p)) &\leq (\sqrt{2(np-2)} + \frac{13}{4})r \quad \forall \quad np \leq 2.5 \\ I\!E(L_{T_{PVCP}}(x^n, p)) &\leq (\frac{np}{2} + 3)r \quad \forall \quad np > 2.5 \end{aligned}$$

Ces bornes sont améliorées dans le cas de suite aléatoire, dans ce cas $\mathbb{E}(L_T(X^n, p))$ est une variable aléatoire respectivement à la position des points.

Proposition 8 *(Jaillet [34])*

$$\begin{aligned} \mathbf{Eu}(I\!E(L_{T_{PVCP}}(X^n, p))) &\leq (\sqrt{\tfrac{4}{3}(np-3)} + \frac{11}{12} + \sqrt{2})r \quad \forall \quad np \leq 3.75 \\ \mathbf{Eu}(I\!E(L_{T_{PVCP}}(X^n, p))) &\leq (\frac{np}{3} + \frac{2}{3} + \sqrt{2})r \quad \forall \quad np > 3.75 \end{aligned}$$

Théorème 2 *(Jaillet [35],[41])*
Soit X une suite de points uniformément et indépendamment distribués dans $[0,1]^2$ *et p la probabilité de présence de chaque point. Alors il existe une constante* $c(p)$ *tel que :*

$$\lim_{n\to\infty} \frac{I\!E(L_{T_{PVCP}}(X^n, p))}{\sqrt{n}} = c(p) \ (p.s) \tag{2.6}$$

Remarques :

- La démonstration est basée sur le résultat très important des fonctions subadditives de *Steele* (voir [60]) :

  Théorème 3 *(Steele) soit* ϕ *une fonction réelle d'un sous ensemble fini de* IR^2 *vérifiant :*

  (a) ϕ *euclidienne : linéaire et invariante par translation.*

  (b) ϕ *monotone :* $\phi(A \cup \{z\}) \geq \phi(A)$ *pour tout sous ensemble fini A de* IR^2

(c) *ϕ a une variance finie, si les points de $X^{(n)}$ sont indépendamment et uniformément distribuées dans $[0,1]^2$: $var[\phi(X^{(n)})] < \infty$*

(d) *ϕ est subadditive : Si $\{Q_i\}_{i=1}^{m^2}$ est une partition de carré $[0,1]^2$ en sous carrés de coté $\frac{1}{m}$ et si $rQ_i \equiv \{\eta : \eta = r\epsilon, \epsilon \in Q_i\}$, alors il existe une constante $c > 0$ tel que pour tout entier positif m et tout réel r on a :*

$$\phi(x(n) \cap [0,r]^2) \leq \sum_{i=1}^{m^2} \phi(x(n) \cap rQ_i) + crm. \tag{2.7}$$

Sous les hypothèses (a),(b),(c),(d) et si X est une suite de points indépendamment et uniformément distribués dans $[0,1]^2$, il existe une constante $\beta(\phi)$ tel que

$$\lim_{n\to\infty} \frac{\phi(X^{(n)})}{\sqrt{n}} = \beta(\phi) \quad (p.s)$$

- Sous les conditions du Théorème 2 on a d'après la Proposition 8 et le Théorème de la convergence dominée :

$$\lim_{n\to\infty} \frac{Eu(\mathbb{E}(L_{T_{PVCP}}(X^n, p)))}{\sqrt{n}} = c(p) \tag{2.8}$$

Théorème 4 *(Jaillet [35],[41]) Soit X une suite de points uniformément et indépendamment distribués dans $[0,1]^2$ et soit p la probabilité de présence de chaque point, alors*

$$\lim_{n\to\infty} \frac{I\!E(\Sigma_{PVC}(X^n, p))}{\sqrt{n}} = \beta\sqrt{p} \ (p.s) \tag{2.9}$$

Remarques :

- Ces résultats ont été généralisés lorsque les points sont répartis dans un ensemble Lebesgue-mesurable quelconque d'un espace euclidien de dimension $d \geq 2$, les constantes ne dépendant que de la dimension de l'espace et de la probabilité de présence des points, mais pas de la forme de l'ensemble.

- La comparaison asymptotique des deux stratégies, de réoptimisation et a priori, reste encore mal comprise.

- Le Théorème 3 de convergence des fonctionnelles de certains POCPs ne donne aucune information sur les valeurs des constantes. On n'a pu déterminer pour le PVCP la valeur de la constante, on se contente de donner des bornes. La meilleur borne pour $c(p)$ est :

Lemme 1 *(Jaillet [35])*

$$\beta\sqrt{p} \leq c(p) \leq \min(\beta, 0.9204\sqrt{p})$$

Après cette présentation des résultats déjà existant, nous allons introduire les résultats significatifs de notre travail.

2.2 Bornes pour le PVCP

Dans [34], on montre l'inégalité (2.4) dans le cas où le nombre de points est premier. En effet, pour n premier chaque T^r est un tour et on a donc $L_{T^r} \geq L_{T^0_{PVC}}$ puisque PVC est le tour optimal. D'après (2.2) on a :

$$\mathbb{E}(L_{T_{PVCP}}) \geq p(1-(1-p)^{n-1})L_{T^0_{PVC}} \quad \text{n premier}$$

On va donner des résultats analogues pour n quelconque. Il est clair que si n n'est pas premier l'analyse ci dessus n'est plus vraie car il existerait des T^r qui ne sont pas des tours. Mais il y aura au moins un tour parmi les T^r, et on va utiliser cette propriété afin de généraliser l'inégalité (2.4). Pour cela on commence par rappeler la définition de structure de cycle d'un tour [5].

Définition 1 *Soit T un tour quelconque on définit C_T la structure de cycle de T :*

$$C_T = (T^0, T^1, ..., T^{n-2})$$

où T^r est l'ensemble de $pgcd(n, r+1)$ sous-tours qui sont formés en sautant r sommets dans T.

L'idée de la démonstration des deux propositions suivantes est qu'on peut toujours trouver au moins un tour, qu'on note T_1, dans la structure de cycle du tour optimal au sens PVCP. Et on montre que la structure du cycle du tour T_1 n'est qu'une simple permutation de la structure du cycle du tour optimal du PVCP, ainsi on peut déduire une comparaison des fonctionnelles des deux tours T_1 et T_{PVCP}.

Proposition 9 *Soit T_{PVCP} le tour optimal du PVCP à travers n sommets, si n impair $(n = 2k+1)$ alors :*

$$\mathbb{E}(L_{T_{PVCP}}) \geq p^2 L_{T^0_{PVCP}} \frac{1-(1-p)^{n-1}}{1-(1-p)^k} \tag{2.10}$$

Démonstration :
On suppose que le graphe est symétrique alors d'après (2.3) pour tout tour T on a :

$$L_{T^r} = L_{T^{n-2-r}} \qquad \text{pour} \qquad 0 \leq r \leq k-1$$

Ainsi la structure de cycle d'un tour T à travers $2k+1$ sommets est :

$$C_T = (T^0, T^1, ...T^{k-1}, T^{k-1}, ..., T^1, T^0)$$

D'après la symétrie et (2.2), l' espérance de T_{PVCP} est :

$$\mathbb{E}(L_{T_{PVCP}}) - p^2(1+q^{n-2})L_{T^0_{PVCP}} = p^2 \sum_{r=0}^{k-2} (q^{r+1} + q^{n-2-r-1}) L_{T^{r+1}_{PVCP}} \tag{2.11}$$

Donc en posant $k_1 = div((k-1),2)$ et $k_2 = k-1-k_1$ on obtient d'après (2.11)

$$\begin{aligned} \mathbb{E}(L_{T_{PVCP}}) - p^2(1+q^{n-2})L_{T^0_{PVCP}} &= p^2 \sum_{r=0}^{k_2-1} (q^{2r+1} + q^{n-2-2r-1}) L_{T^{2r+1}_{PVCP}} \\ &+ p^2 \sum_{r=0}^{k_1-1} (q^{2(r+1)} + q^{n-2-2(r+1)}) L_{T^{2(r+1)}_{PVCP}} \end{aligned} \tag{2.12}$$

Comme n est impair $pgcd(n,2) = 1$ donc T^1_{PVCP} est un tour, on pose

$$T_1 = T^1_{PVCP}.$$

On remarque que la structure de cycle de T_1 se déduit de la structure de cycle de T_{PVCP} :

$$T_1^r = T^{2r+1}_{PVCP} \quad \forall\, r \quad 0 \leq r \leq k-1 \tag{2.13}$$

En effet T_1^r est l'ensemble de $pgcd(n, r+1)$ sous-tours qui sont formés en sautant r sommets de T_1 on remarque que :

- sauter r sommets de T_1 revient à sauter $2r+1$ sommets de T_{PVCP} (car T_1 est le tour formé en sautant un sommet de T_{PVCP})
- $pgcd(n,(2r+1)+1) = pgcd(n, r+1) \quad \forall\, r \quad 0 \leq r \leq k-1$ car n est impair.

D'après les remarques ci dessus, T_1^r est l'ensemble de $pgcd(n, 2r+2)$ sous-tours qui sont formés en sautant $2r+1$ sommets de T_{PVCP} d'ou (2.13).
Ainsi d'après la symétrie, (2.2) et (2.13) on obtient :

$$\begin{aligned} \mathbb{E}(L_{T_1}) - p^2(q^{k-1}+q^k)L_{T^0_{PVCP}} &= p^2 \sum_{r=0}^{k_2-1} (q^r + q^{n-2-r}) L_{T^{2r+1}_{PVCP}} \\ &+ p^2 \sum_{r=0}^{k_1-1} (q^{k-2-r} + q^{n-2-(k-2-r)}) L_{T^{2(r+1)}_{PVCP}} \end{aligned} \tag{2.14}$$

En remarquant que :

$$\begin{cases} q^k(q^r + q^{n-2-r}) \leq q^{2r+1} + q^{n-2-(2r+1)} & \forall\, 0 \leq r \leq k_1 - 1. \\ q^k(q^{k-2-r} + q^{k-1-r}) \leq q^{2(r+1)} + q^{n-2-2(r+1)} & \forall\, 0 \leq r \leq k_1 - 1 \end{cases} \tag{2.15}$$

et en multipliant l'équation (2.14) par q^k, on obtient, d'après l'équation (2.12), l'inégalité suivante :

$$q^k[\mathbb{E}(L_{T_1}) - p^2(q^{k-1}+q^k)L_{T^0_{PVCP}}] \leq \mathbb{E}(L_{T_{PVCP}}) - p^2(1+q^{n-2})L_{T^0_{PVCP}} \tag{2.16}$$

Comme T_1 est un tour et T_{PVCP} est le tour optimal au sens PVCP on a

$$\mathbb{E}(L_{T_{PVCP}}) \leq \mathbb{E}(L_{T_1}) \tag{2.17}$$

On trouve alors d'après (2.16) et (2.17)

$$p^2(1+q^{n-2} - q^k(q^{k-1}+q^k))L_{T^0_{PVCP}} \leq (1-q^k)\mathbb{E}(L_{T_{PVCP}})$$

Donc

$$p^2 \frac{1-q^{n-1}}{1-q^k} L_{T^0_{PVCP}} \leq \mathbb{E}(L_{T_{PVCP}})$$

∎

Remarques :

- D'après l'égalité (2.10) on déduit :

$$\frac{L_{T^0_{PVCP}} - L_{T^0_{PVC}}}{L_{T^0_{PVC}}} \leq \frac{1-p^2}{p^2}$$

En effet :

$$\mathbb{E}(L_{T_{PVC}}) - \mathbb{E}(L_{T_{PVCP}}) \geq 0.$$

alors

$$L_{T^0_{PVC}} - p^2 \frac{1-q^{n-1}}{1-q^k} L_{T^0_{PVCP}} \geq 0.$$

donc

$$\frac{L_{T^0_{PVCP}} - L_{T^0_{PVC}}}{L_{T^0_{PVC}}} \leq \frac{1-p^2}{p^2}$$

- De même on obtient

$$\frac{\mathbb{E}(L_{T_{PVC}}) - \mathbb{E}(L_{T_{PVCP}})}{\mathbb{E}(L_{T_{PVCP}})} \leq \frac{1-p^2}{p^2}$$

- La borne (2.10) est moins bonne que celle obtenue dans le cas n premier.
- La démonstration reste la même dans le cas non symétrique, il suffit de remplacer q^k par q^n.

On a utilisé dans la démonstration précédente le fait que T^1_{PVCP} est un tour, ceci n'est plus vrai pour n pair. Or dans le cas pair on sait qu'on a un tour dans la structure de cycle : pour tout $n > 6$, on note $r_0 + 1$ le premier nombre premier avec n alors la permutation $T^{r_0}_{PVCP}$ est un tour et $r_0 < n/2$.
Nous remarquons aussi que pour un tour T, si T^{k_0} induit de T en sautant k_0 sommets de T est un tour, qu'on notera T_{k_0}, alors la structure de cycle de T_{k_0} se déduit de celle de T :

$$T^r_{k_0} = T^{(k_0(r+1)+r) \bmod n} \qquad \forall\, r \tag{2.18}$$

En effet $T^r_{k_0}$ est l'ensemble de $pgcd(n, r+1)$ sous tours en sautant r sommets de T_{k_0}. Or d'après la définition de T_{k_0} (T_{k_0} est le tour résultant de T en sautant k_0 sommets), sauter r sommets de T_{k_0} revient à sauter $k_0(r+1)+r$ sommets de T. T_{k_0} étant un tour $pgcd(n, k_0+1) = 1$ d'ou

$$pgcd(n, (k_0(r+1)+r)+1) = pgcd(n, (k_0+1)(r+1)) = pgcd(n, r+1)$$

donc $T^r_{k_0} = T^{(k_0(r+1)+r) \bmod n}$.

Ces deux remarques nous ont conduit à reprendre la démonstration de la proposition précédentes afin de généraliser le résultat au cas de n pair.

Proposition 10 *Soit T_{PVCP} le tour optimal du PVCP à travers $n = 2k > 6$ sommets alors on a*

$$I\!E(L_{T_{PVCP}}) \geq p^2 L_{T^0_{PVCP}} \frac{1-(1-p)^{n+h_{r_0}}}{1-(1-p)^n} \tag{2.19}$$

et :$pgcd(n, r_0+1) = 1 \Rightarrow \exists\ (h_n, h_{r_0})\ /\ nh_n + (r_0+1)h_{r_0} = 1$.

Démonstration :
Comme $n = 2k$, la structure de cycle de T, C_T est :

$$C_T = (T^0, T^1, ..., T^{k-2}, T^{k-1}, T^{k-2}, ..., T^1, T^0) \tag{2.20}$$

D'après (2.20) et la définition de $\mathbb{E}(L_T)$ on a pour tout T

$$\mathbb{E}(L_T) - p^2(1+q^{n-2})L_{T^0} = p^2 \sum_{r=1}^{k-2}(q^r + q^{n-2-r})L_{T^r} + p^2 q^{k-1} L_{T^{k-1}} \tag{2.21}$$

D'après (2.18) on a

- $L^{k-1}_{T_{r_0}} = L^{k-1}_{T_{PVCP}}$
- Comme $pgcd(n, r_0+1) = 1$ donc $\exists\ (h_n, h_{r_0})\ /\ nh_n + (r_0+1)h_{r_0} = 1$. alors

$$L_{T^{h_{r_0}}_{r_0}} = L_{T^0_{PVCP}} \tag{2.22}$$

On a d'après (2.21) et (2.22)

$$\begin{aligned} \mathbb{E}(L_{T_{r_0}}) - p^2(q^{h_{r_0}} + q^{n-2-h_{r_0}})L_{T_{PVCP}} &= p^2 \sum_{\substack{r=0 \\ r\neq h_{r_0}}}^{k-2} (q^r + q^{n-2-r}) L_{T^{(r_0(r+1)+r) \bmod n}_{PVCP}} \\ &+ p^2 q^{k-1} L^{k-1}_{T_{PVCP}} \end{aligned} \tag{2.23}$$

En remarquant que :

$$q^n(q^r + q^{n-2-r}) \leq (q^{(r_0(r+1)+r) \bmod n} + q^{n-2-(r_0(r+1)+r) \bmod n})$$

et en multipliant l'équation (2.23) par q^n, d'après l'équation (2.21) pour le tour T_{PVCP},

$$q^n[\mathbb{E}(L_{T_{r_0}}) - p^2(q^{h_{r_0}} + q^{n-2-h_{r_0}})L_{T_{PVCP}}] \leq \mathbb{E}(L_{T_{PVCP}}) - p^2(1+q^{n-2})L_{T^0_{PVCP}} \tag{2.24}$$

On déduit de la même façon que la proposition (9)que :

$$\mathbb{E}(L_{T_{PVCP}}) \geq p^2 L_{T^0_{PVCP}} \frac{1-q^{n+h_{r_0}}}{1-q^n}$$

∎

2.3 Cas particuliers

Le $PVCP$ étant $NP-$dur, nous allons chercher des cas particuliers où le problème est résoluble en un temps polynomial.
On sait que sous certaines conditions, sur la matrice distance, (qu'on notera C dans ce paragraphe) le PVC est résoluble en temps polynomial [49]. Nous allons montrer que dans certains de ces cas le $PVCP$ est équivalent au PVC.

$PVCP$ constant

Nous allons donner des conditions sur la matrice distance C sous lesquelles tous les tours *hamiltoniens* ont la même espérance. Berenguer [4] a montré le théorème suivant :

Théorème 5 *(Berenguer) Les matrices de la forme $c_{ij} = a_i + b_j$ sont les seules à vérifier la propriété suivante :*
$\mathcal{P}$: Toutes les permutations ont la même longueur.

Les matrices C de la forme $c_{ij} = a_i + b_j$ sont appelées matrices constantes.

Proposition 11 *Les matrices de la forme $c_{ij} = a_i + b_j$ sont les seules à avoir la même espérance pour tout tour :*

$$\mathbb{E}(L_T) = p(1-(1-p)^{n-1})L_{T^0_{PVC}} \quad \forall \quad T.$$

Démonstration : Soit T un tour quelconque alors :

$$\mathbb{E}(L_T) = p^2 \sum_{r=0}^{n-2} (1-p)^r L_{T^r}$$

avec

$$L_{T^r} = \sum_{i=1}^{n} d(i, T^r(i))$$

Puisque T^r est une permutation, d'après le Théorème 5 on a :

$$L_{T^r} = L_{T^0_{PVC}} \quad \forall\, T \qquad \forall\, r.$$

Donc

$$\mathbb{E}(L_T) = p(1-(1-p)^{n-1})L_{T^0_{PVC}} \quad \forall \quad T.$$

Il reste à montrer que si on a la même espérance pour tout tour alors la matrice distance est constante. Supposons qu'il existe une matrice non constante qui vérifie

$$\mathbb{E}(L_T) = p(1-(1-p)^{n-1})L_{T^0_{PVC}} \quad \forall \quad T$$

D'après le théorème de Berenguer il existe T, r tel que.

$$L_{T^r} > L_{T^0_{PVC}} \tag{2.25}$$

Donc il $\exists$ T tel que

$$\mathbb{E}(L_T) > p(1-(1-p)^{n-1})L_{T^0_{PVC}} \tag{2.26}$$

∎

Remarques :

- Pour les graphes ayant une matrice coût constante n'importe quel tour résout le $PVCP$, donc la résolution du $PVCP$ se fait en un temps polynomial.
- Pour les graphes ayant une matrice constante, la "stratégie a priori" est équivalente à la "stratégie de ré-optimisation", car toute sous-matrice d'une matrice constante est une matrice constante et donc tout tour est une solution optimale pour chaque exemplaire.
- La proposition (11) se généralise pour une loi de probabilité quelconque.

$PVCP$ petit

Définition et notation On dit que C est une petite matrice s'il existe deux vecteurs a et b tel que $c_{i,j} = min\{a_i, b_j\}$. Une petite matrice où les a_i et b_j sont distincts est dite à valeur distinctes. Soit d_i le $i^{\text{ème}}$ plus petit des $2n$ valeurs a_i et b_j, on note $D = \{d_1, ...d_n\}$, $\bar{D} = \{d_{n+1}, ...d_{2n}\}$ et $d = \sum_{i=1}^{n} d_i$.

On pose :

$$\begin{aligned} D_2 &= \{i \text{ tel que } a_i \text{ et } b_i \in D\} \\ D_0 &= \{i \text{ tel que } a_i \text{ et } b_i \in \bar{D}\} \\ D_a &= \{i \text{ tel que } a_i \in D \text{ et } b_i \in \bar{D}\} \\ D_b &= \{i \text{ tel que } b_i \in D \text{ et } a_i \in \bar{D}\} \end{aligned}$$

On rappelle les théorèmes de Gabovitch 1970 (voir [24] ou [49] p 92), où il montre que pour les petites matrices le PVC est un problème facile.

Théorème 6 *(Gabovitch) Soit C une petite matrice à valeur distinctes la longueur du tour optimal pour C est d si et seulement si une des conditions suivantes est vérifiée :*

(S1) $D_2 \neq \emptyset$

(S2) $D = \{a_1, ..., a_n\}$.

(S3) $D = \{b_1, ..., b_n\}$.

Remarques :

- Il est clair que $|D_2| = |D_0|$

- Si l'une des conditions $(S1)$, $(S2)$ ou $(S3)$ est vérifiée alors le tour optimal est construit comme suit [49] : on commence par un sommet de D_2, puis on visite tous les sommets de D_a dans n'importe quel ordre, ensuite, on visite un sommet de D_0, puis tous les sommets de D_b dans n'importe quel ordre, on termine le tour en visitant alternativement un sommet de D_2 et de D_0. Si on note 1_{D_i} un élément de D_i pour $i \in \{0,2\}$ on a alors

$$T_{PVC} = (1_{D_2}, D_a, 1_{D_0}, D_b, 1_{D_2}, 1_{D_0}, \ldots, 1_{D_2}, 1_{D_0}) \tag{2.27}$$

  D'après cette construction on remarque que $c_{iT_{PVC}(i)} \in D$ pour tout i.

- Si $(S1)$, $(S2)$ et $(S3)$ ne sont pas vérifiées alors il existe k, $1 \leq k \leq n-1$, tel que :

$$D = \{a_1, ..., a_k, b_{k+1}, ...b_n\}$$

  ainsi on ne peut pas construire un tour de coût d : en effet si on a un tour de coût d alors il existe un arc dans T de coût $b_i \in D$ suivit par un arc de coût $a_j \in D$ et donc $i = j$ (absurde car T est un tour).

- Ces remarques donnent le schéma de la preuve du Théorème 6.

Par une démonstration presque identique du Théorème 6 Gabovitch donne une généralisation, pour cela on va introduire les notations suivantes. On dit qu'un ensemble de coût $D = \{d_1, ..., d_n\}$ domine un autre ensemble de coût $D' = \{d'_1, ..., d'_n\}$ s'il existe une permutation σ tel que $d'_i \leq d_{\sigma(i)}$ pour $i \in [1..n]$. On dit que l'ensemble de coût $D = \{d_1, ...d_n\}$ est réalisable s'il existe un tour T et une permutation σ tel que $c_{iT(i)} = d_{\sigma(i)}$ pour $i \in [1..n]$.

Théorème 7 *(Gabovitch) Soit C une petite matrice à valeurs distinctes, $D' = \{d'_1, ...d'_n\}$ un sous-ensemble de $\{a_1, ..., a_n, b_1, ..., b_n\}$ tel que tout sous-ensemble que domine D' n'est pas réalisable. Alors il existe un tour de coût $\sum_{i=1}^{n} d'_i$ si et seulement si une des conditions suivantes est vérifiée :*

(S1') $D'_2 \neq \emptyset$

(S2') $D' = \{a_1, ..., a_n\}$.

(S3') $D' = \{b_1, ..., b_n\}$.

Donc si D n'est pas réalisable qu'elle est le coût du tour optimal ? le théorème suivant caractérise le tour optimal pour une petite matrice.

Théorème 8 *(Gabovitch) Soit C une petite matrice à valeurs distinctes. La longueur du tour optimal est soit d, $d - d_n + d_{n+1}$ ou $min(d - d_n + d_{n+2}, d - d_{n-1} + d_{n+1})$. Si le coût du tour optimal n'est pas d alors le coût du tour optimal est supérieur à $d - d_n + d_{n+1}$ si et seulement si une des conditions suivantes est vérifiée :*

(S4) $k = 1; d_n = b_2; d_{n+1} = a_2$.

(S5) $k = n-1; d_n = a_{n-1}; d_{n+1} = b_{n-1}$.

(S6) $2 \leq k \leq n-2$ *; alors soit* $(d_n = a_k$ *et* $d_{n+1} = b_k)$ *ou* $(d_n = b_{k+1}$ *et* $d_{n+1} = a_{k+1})$.
On rappelle que k *est le plus grand indice* i *tel que* $a_i \in D$

L'idée de la démonstration du théorème 8 est la suivante : si les hypothèses du théorème 6 ((*S1*) (*S2*) et (*S3*)) ne sont pas vérifiées alors D n'est pas réalisable. Il est clair que l'ensemble de coût $D' = D \cap \{d_{n+1}\} - \{d_n\}$ domine D et alors d'après le théorème 7 il existe un tour de coût $d - d_n + d_{n+1}$ ssi D' satisfait (*S1'*), (*S2'*) ou (*S3'*). Si (*S1'*), (*S2'*) et (*S3'*) ne sont pas vérifiées pour D' alors $d - d_n + d_{n+1}$ n'est pas réalisable, on peut montrer que les deux ensembles $D \cap \{d_{n+2}\} - \{d_n\}$ et $D \cap \{d_{n+1}\} - \{d_{n-1}\}$ doivent satisfaire une des conditions du théorème 7, ces des ensembles sont réalisables et la solution optimale est donc le minimum de deux valeurs $\min\{d - d_{n+2} + d_{n+1}, d - d_{n-1} + d_{n+1}\}$.

D'après les théorèmes ci dessus nous allons déduire certains résultats pour le *PVCP*. Plusieurs cas se posent :

Corollaire 1 *Soit C une petite matrice à valeurs distinctes. L'espérance du tour optimal (au sens PVCP) est* $p(1-q^{n-1})d$, *si et seulement si (S2) ou (S3) est vérifiée.*

Démonstration : Une petite matrice est une matrice constante si et seulement si (S2) ou (S3) est vérifié, d'où le résultat d'après la proposition 11. ∎

Corollaire 2 *Soit C une petite matrice à valeurs distinctes. Si (S1), (S2) et (S3) ne sont pas vérifiées alors l'espérance du tour optimal (au sens PVCP) est égale à* $p(1-q^{n-1})(d - d_n + d_{n+1})$ *si et seulement si* $D' = D \cap \{d_{n+1}\} - \{d_n\}$ *vérifie (S2') ou (S3').*

Corollaire 3 *Soit C une petite matrice à valeurs distinctes. Si (S1), (S2) et (S3) ne sont pas vérifiées et si* $D' = D \cap \{d_{n+1}\} - \{d_n\}$ *ne vérifie pas (S1'), (S2') et (S3'), alors PVCP* $\equiv$ *PVC et l'espérance du tour optimal (au sens PVCP) est* $p(1-q^{n-1})\min\{d - d_{n+2} + d_n, d - d_{n-1} + d_{n+1}\}$ *si et seulement si*

(S4') $D \cap \{d_{n+2}\} - \{d_n\}$ *satisfait (S2') ou (S3') et* $(d + d_{n+2} - d_n \leq d - d_{n-1} + d_{n+1})$.

(S5') $D \cap \{d_{n+1}\} - \{d_{n-1}\}$ *satisfait (S2') ou (S3') et* $(d + d_{n+2} - d_n \geq d - d_{n-1} + d_{n+1})$.

On peut construire des exemples où le *PVCP*$\neq$ *PVC*. Nous allons donner des conditions pour que la stratégie de *PVCP* et de ré-optimisation soient équivalentes. Pour cela nous allons montrer que sous certaines conditions la solution est stable : c'est-à-dire si S est l'ensemble de sommets présents, le tour induit du T_{PVC} en sautant les sommets qui ne sont pas dans S est un tour optimal à travers S. En effet d'après les théorèmes de Gabovitch on peut construire la solution du PVC pour une petite matrice (par caractérisation de D_2,D_0,D_a et D_b). Du fait qu'une sous-matrice d'une petite matrice est une petite-matrice alors il est possible toujours d'après les théorèmes de Gabovitch de construire les solutions optimales au sens PVC pour chaque exemplaire S. Ainsi dans nos démonstration nous vérifiant que pour chaque exemplaire S le tour induit du tour optimal est un tour optimal à travers S.

On se restreint au cas (S1) vérifiée et $(|D_2| = 1, D_b = \emptyset)$: on sait que $|D_2| = |D_0|$, on suppose que $D_2 = \{1\}$ *et* $D_0 = \{n\}$. On pose $d' = d + d_{n+1} - d_n$.

Proposition 12 *Soit C une petite matrice à valeurs distinctes. Si (S1) est vérifiée et* ($|D_2| = 1, D_b = \emptyset$) *alors PVCP ≡ PVC si et seulement si une des conditions suivantes est vérifiée :*

(C1) $[(d_n = b_1)$ *ou* $(d_n = a_1$ *et* $d_{n-1} = b_1)]$ *et* $[(d_{n+1} = a_n)$ *ou* $(d_{n+1} = b_n$ *et* $d_{n+2} = a_n)]$

On a :

$$I\!E(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})d' - p^2(d_{n+1} - d_n)$$

Démonstration : D'après (2.27) le tour optimal T_{PVC} est :

$$T_{PVC} = \{1\} \cup D_a \cup \{n\}$$

Nous allons montrer que sous la condition (C1) la solution est stable : c'est-à-dire si S est l'ensemble de sommets présents, le tour induit du T_{PVC} en sautant les sommets qui ne sont pas dans S est un tour optimal à travers S. On note $T_{PVC}(S)$ le tour induit de PVC à travers S.

On remarque que les sous-matrices de petite matrice sont des petites matrices. D'après le Théorème 8 on peut déterminer le tour optimal pour tout sous-problème. Nous allons montrer que sous la condition (C1) tous les sous-problèmes vérifient l'une des conditions du Théorème 8, on déduit alors pour tout $S \subset \{1, ..., n\}$ le tour optimal à travers S et on vérifie que c'est le tour induit du PVC à travers S en sautant les sommets absents.

On note d_i^S le $i^{\text{ème}}$ plus petit des $2|S|$ valeurs a_j et b_j dans S, on pose $D^S = \{d_i^S\}_{1 \le i \le |S|}$, $D_a^S = \{i \in S | a_i \in D^S$ et $b_i \notin D^S\}$

On remarque qu'on a quatre catégories de sous ensembles :

* Si $S \subset D_a$ (1 et n absents)

  alors D^S vérifie (S2) donc tout tour est optimal.

* Si $S = \{1\} \cup \mathcal{P}(D_a) \cup \{n\}$ (1 et n présents)

  alors D^S vérifie (S1) avec $S \cap D_2 = \{1\}$ et $S \cap D_0 = \{n\}$ donc d'après (2.27) le tour induit de PVC est le tour optimal à travers S.

* Si $S = \{1\} \cup \mathcal{P}(D_a)$ (1 présent et n absent)

  alors

  $$D^S = \{a_1, b_1\} \cup D_a^S \setminus \max\{a_1, b_1\} \cup D_a^S$$

  D'après $(C1)$ on a :

  $$\max\{\{a_1, b_1\} \cup D_a^S\} = \max\{a_1, b_1\}$$

  On déduit donc deux cas

  ** Si $d_n = b_1$ donc $D^S = D_a^S \cup \{a_1\}$ d'ou D^S vérifie (S2).

  ** Sinon $d_n = a_1$ et $d_{n-1} = b_1$ alors

  $$D^S = D_a^S \cup \{b_1\}$$

  dans ce cas D^S ne vérifie aucune des conditions du théorème 6 donc D^S n'est pas réalisable. On considère

  $$D'^S = D^S \cup \min\{a_1, \{b_i\}_{i \in S}\} \setminus \max D^S$$

sachant que $d_n = a_1$ et $d_{n-1} = b_1$ on a $\max D^S = b_1$ et $\min\{a_1, \{b_i\}_{i\in S}\} = a_1$ et donc

$$D'^S = D^S \cup \{a_1\} \backslash \{b_1\}$$

D'^S vérifie (S2').

* Si $S \subset \mathcal{P}(D_a) \cup \{n\}$ (n présent et 1 absent)

$$D^S = D_a^S \cup \min\{a_n, b_n, \{b_i\}_{i\in S}\}$$

D'après $(C1)$ on a deux cas :

** Si $d_{n+1} = a_n$ alors D^S vérifie (S2).

** Sinon $d_{n+1} = b_n$ et $d_{n+2} = a_n$ alors D^S ne vérifie aucune condition du théorème 6. On considère

$$D'^S = D^S \cup \min\{a_n, b_n, \{b_i\}_{i\in S}\} \backslash \max D^S$$

comme $\max D^S = b_n$ et $\min\{a_n, \{b_i\}_{i\in S}\} = a_n$ donc D'^S vérifie (S2').

Calcul de l'espérance du $PVCP$:

$$\mathbb{E}(L_{T_{PVCP}}) = p^2 \sum_{r=0}^{n-2} (1-p)^r L_{T^r_{PVCP}}$$

comme $PVCP = PVC = \{1\} \cup \{2, 3, 4, .., n-1\} \cup \{n\}$ alors :

$$L_{T^0_{PVC}} = d$$

et

$$L_{T^r_{PVC}} = \sum_{i=1}^{n} \min(a_i, b_{i+1+r}) = \sum_{i=1}^{n-1} a_i + \min(a_n, b_{1+r}) \quad \forall\, r \geq 1$$

Or d'après la condition $(C1)$ $\min(a_n, b_{1+r}) = a_n$ on a :

$$L_{T^r_{PVC}} = d + d_{n+1} - d_n = d' \quad \forall\, r \geq 1$$

Donc

$$\mathbb{E}(L_{T_{PVCP}}) = p^2(d_n - d_{n-1}) + p(1 - q^{n-1})d'$$

∎

Sous la condition $(S1)$, $(S2)$ et $(S3)$ ne sont pas vérifiées, si la condition (C1) n'est pas vérifiée alors les tours optimaux au sens PVC ont des espérances différentes. Nous allons chercher ceux qui sont d'espérance minimale (au sens $PVCP$). On pose

$$T_{PVC_{D_a}} = \{1\} \cup \{2, 3, 4, .., n-1\} \cup \{n\} \text{ tel que } a_2 \leq a_3 ... \leq a_{n-1}.$$

Les démonstrations des propositions suivantes se font de la même façon que la proposition 12, on donnera donc que les détailles du calcul de l'espérance.

Proposition 13 *Soit C une petite matrice à valeurs distinctes qui vérifie (S1) et $(|D_2| = 1, D_b = \emptyset)$ alors $T_{PVCP} = T_{PVC_{D_a}}$ si une des conditions suivantes est vérifiée*

(C2) $[d_n \neq b_1]$ *et* $[(d_n \neq a_1)$ *ou* $(d_{n-1} \neq b_1))]$ *et* $[(d_{n+1} = a_n)$ *ou* $((d_{n+1} = b_n)$ *et* $(d_{n+2} = a_n))]$

On a

$$I\!E(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})(d+a_n) - p^2 \sum_{r=0}^{n-2} q^r max(a_{n-r}, b_1)$$

Démonstration : Calcul de l'espérance du $PVCP$:

$$L_{T^0_{PVCP}} = L_{T^0_{PVC}} = d$$

sous les conditions on a

$$L^{(r)}_{T^0_{PVC}} = d + min(a_n, b_{1+r}) - max(a_{n-r}, b_1)$$

Donc

$$I\!E(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})(d+a_n) - p^2 \sum_{r=0}^{n-2} q^r max(a_{n-r}, b_1)$$

∎

On note $T_{PVC_{D_a}}$ le tour optimal ordonné suivant les b_i tel que $i \in D_a$ donc

$$T_{PVC_{D_a}} = \{1\} \cup \{2,3,4,..,n-1\} \cup \{n\} \text{ tel que } b_2 \leq b_3 ... \leq b_{n-1}.$$

Proposition 14 *Soit C une petite matrice à valeurs distinctes qui vérifie (S1) et $(|D_2| = 1, D_b = \emptyset)$ alors $T_{PVCP} = T_{PVC_{D_a}}$ pour tout n si une des conditions suivantes est vérifiée*

(C3) $[(d_n = b_1)$ *ou* $((d_n = a_1)$ *et* $(d_{n-1} = b_1))]$ *et* $[d_{n+1} \neq a_n]$ *et* $[(d_{n+1} \neq b_n)$ *ou* $(d_{n+2} \neq a_n)]$

On a

$$I\!E(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})(d-b_1) + p^2 \sum_{r=0}^{n-2} q^r min(a_n, b_{1+r}).$$

Démonstration : Calcul de l'espérance du $PVCP$:

$$L_{T^0_{PVC}} = L_{T^0_{PVC}} = d$$

Sous la condition (C3) on a :

$$L_{T^r_{PVC}} = d + min(a_n, b_{1+r}) - b_1$$

Donc

$$I\!E(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})(d-b_1) + p^2 \sum_{r=0}^{n-2} q^r min(a_n, b_{1+r})$$

∎

Remarques :

- Si les conditions des deux propositions précédentes sont vérifiées, et
  (C4) ($a_2 \leq ... \leq a_{n-1}$ et $b_2 \leq ... \leq b_{n-1}$) est vérifiée alors

$$T_{PVCP} = T_{PVC_{D_a}} = T_{PVC_{D_a}}.$$

  Dans ce cas

$$\mathbb{E}(L_{T_{PVCP}}) = p(1-(1-p)^{n-1})d + p^2 \sum_{r=0}^{n-2} q^r(\min(a_n, b_{1+r}) - \max(a_{n-r}, b_1))$$

- Si $(C1)$, $(C2)$, $(C3)$ et $(C4)$ ne sont pas vérifiées alors il existerait au moins un sous-ensemble tel que le tour induit du tour optimal PVC n'est pas le tour optimal à travers S donc les stratégies de ré-optimisation et a priori sont différentes.

- Les théorèmes ont leurs équivalents dans des cas plus généraux mais les conditions sont plus lourdes.

- On remarque que les conditions que nous imposons nous permettent de construire le T_{PVC} par un algorithme imperturbable par l'absence des sommets.

- Les proposition précédentes se généralisent pour une loi de probabilité quelconque.

Matrice sur-triangulaire

On dit que C est sur-triangulaire si

$$\forall\,(i,j)\ /\ i \geq j\ c_{ij} = 0.$$

Lawler montre que pour les matrices sur-triangulaires le PVC est aussi facile que le problème de transport (relaxation du PVC : recherche d'une permutation de longueur minimal) [49].

Théorème 9 *(Lawler) Soit C une matrice sur-triangulaire et ϕ la solution du problème de transport optimal sous la contrainte $\phi(n) = 1$ alors le coût de ϕ est égal à la longueur du tour optimal qu'on peut construire à partir de ϕ.*

Remarques :

- D'après le Théorème 9, l'inégalité (2.4) dans le cas de matrice sur-triangulaire est vérifiée pour tout n si la solution du problème de transport optimal vérifie la contrainte $\phi(n) = 1$.

- On peut construire un problème ayant une matrice sur-triangulaire tel que la stratégie de ré-optimisation et la stratégie du $PVCP$ sont différentes.

- Si C est non-négative alors la longueur du plus court chemin entre 1 et n est égale à la longueur du tour optimal [49]. Nous allons montrer que sous certaines conditions le $PVCP$ est aussi facile que le problème du plus court chemin.

Proposition 15 *Soit C une matrice sur-triangulaire non négative. si 1 et n sont présents et le plus court chemin entre 1 et n est* $(1,n)$ *alors*

$$PVCP \equiv PVC$$

Démonstration :

Nous allons montrer que sous ces deux conditions la stratégie de ré-optimisation est la stratégie du $PVCP$:

1. 1 et n sont toujours présents
2. Le plus court chemin entre 1 et n est $(1,n)$

D'après le Théorème de Lawler, comme le plus court chemin entre 1 et n est $(1,n)$ et $c_{i,i-1} = 0$ alors

$$T_{PVC} = \{1, n, n-1, n-2, ..., 2, 1\}$$

Soit S un sous-ensemble de $\{2, ...n-1\}$ de cardinal k, on note $S = \{s_1, ..., s_k\}$ avec $s_1 \leq s_2 .... \leq s_k$, la matrice associée $C(S)$ est une matrice sur-triangulaire non négative et le plus court chemin entre 1 et n quand $\{s_1, ..., s_k\}$ sont présents est $(1,n)$ ceci est une conséquence du principe d'optimalité du plus court chemin : En effet sinon il existerait un chemin entre 1 et n de coût inférieur quand tous les sommets sont présents. Donc on peut appliquer le théorème de Lawler et le tour optimal à travers $(1,n) \cup S$ à la forme suivante :

$$T^S_{PVC} = (1, n, s_k, ...s_1) \equiv T_{PVC}(S) \quad \forall \quad S$$

∎

Proposition 16 *Soit C une matrice sur-triangulaire non négative et pour* $i < j$ *on a* $c_{i,j} \leq c_{k,j}$ $\forall k \; i+1 \leq k \leq j-1$ *alors*

$$PVCP \equiv PVC$$

Démonstration : On remarque que le plus court chemin de 1 à n est $(1,n)$: en effet si on note $PCC(i)$ le plus court chemin de 1 à i, on a d'après le principe d'optimalité :

$$\begin{aligned} PCC(n) &= \min_{1 \leq k \leq n-1} (PCC(k) + c_{k,n}) \\ &= \min\{\min_{2 \leq k \leq n-1} (PCC(k) + c_{k,n}), c_{1,n}\} \end{aligned}$$

Alors si le plus court chemin n'est pas $(1,n)$, il existe k_0, tel que

$$PCC(k_0) + c_{k_0,n} \leq c_{1,n} \quad 2 \leq k_0 \leq n-1$$

ceci est absurde car $c_{1,n} \leq c_{k_0,n}$ et la matrice est non négative.

Donc d'après le Théorème 9 de Lawler

$$T_{PVC} = (1, n, n-1, ..., 2, 1)$$

Soit S un sous-ensemble de $\{1, ..., n\}$ de cardinal k. on note

$$S = \{s_1, ..., s_k\} \text{ avec } s_1 \leq s_2 .... \leq s_k.$$

La sous-matrice $C(S)$ est une matrice sur-triangulaire non-négative gradée et vérifie $c_{s_1,s_k} \leq c_{s,s_k}$ pour $s_1 < s < s_k$, on montre de la même façon que le plus court chemin entre s_1 et s_k est (s_1, s_k). D'après le Théorème 9 de Lawler on a :

$$T^S_{PVC} = (s_1, s_k, ..., s_1) = T_{PVC}(S) \quad \forall \quad S$$

■

Tour pyramidal

On dit qu'un tour T est pyramidal si pour chaque j, $1 \leq j \leq n$.

$$T^{-1}(j) < j < T(j) \text{ ou } T(j) < j < T^{-1}(j).$$

En d'autre terme T est pyramidal s'il est de la forme suivante

$$T = (1, i_1, .., i_r, n, j_1, ..., j_{n-r-2}) \text{ avec } i_1 < ... < i_r \text{ et } j_1 > ... > j_{n-r-2}.$$

Une caractérisation des tours pyramidaux est la suivante. Soit Δ_n l'ensemble de tous les tours pyramidaux à travers n sommets. Alors $\Delta_2 = \{(1,2)\}$ et Δ_{n+1} contient toutes les permutations de la forme $(n, n+1)T$ ou $T(n, n+1)$ où $T \in \Delta_n$ et seulement ces permutations.

Pour chaque matrice C il est possible de calculer le plus petit tour pyramidal grâce à la programmation dynamique. Soit $C(i, j)$ la longueur du plus court chemin hamiltonien de i à j à travers $1, 2..., \max\{i, j\}$, sous la condition que le chemin passe à travers les sommets dans un ordre décroissant de i à 1 après à travers le complémentaire dans un ordre croissant de 1 à j. D'après le principe d'optimalité, on a :

$$C(i,j) = \begin{cases} C(i, j-1) + c_{j-1,j} & \text{pour } i < j-1 \\ \min\limits_{k<i}\{C(i,k) + c_{k,j}\} & \text{pour } i = j-1 \\ C(i-1, j) + c_{i,i-1} & \text{pour } i > j+1 \\ \min\limits_{k<j}\{C(k,j) + c_{i,k}\} & \text{pour } i = j+1 \end{cases}$$

Il est possible de calculer $C(i, n)$ et $C(n, i)$ pour tout $i < n$ en $0(n^2)$ opérations, on commence par les conditions initiales $C(1,2) = c_{12}$ et $C(2,1) = c_{21}$. la longueur du plus petit tour pyramidal est alors donnée par la formule suivante :

$$\min\{C(n-1, n) + c_{n,n-1}, C(n, n-1) + c_{n-1,n}\}$$

Considérons des conditions sous lesquelles on peut trouver un tour optimal au sens PVC qui est pyramidal. Pour une matrice C, on pose

$$d_{ij} = c_{ij} + c_{i+1,j-1} - c_{i,j-1} - c_{i+1,j} \tag{2.28}$$

où par définition $c_{ij} = 0$ si $i = n+1$ ou $j = 0$. Il est facile de voir que

$$c_{ij} = \sum_{k=i}^{n} \sum_{l=1}^{j} d_{kl} \tag{2.29}$$

Si la matrice $D = (d_{ij})$, définit par (2.28), est non négative, on dit que C est une matrice de distribution générée par la matrice densité D. On rappelle le Lemme [49] qui se démontre facilement d'après l'égalité (2.29).

Lemme 2 *Si C est une matrice de distribution alors*

$$c_{i'j'} + c_{ij} \leq c_{ij'} + c_{i'j}$$

Pour tout $i < i'$ et $j < j'$

Dans le cas où C est une matrice de distribution, on peut donner une borne inférieure pour $\mathbb{E}(L_{T_{PVCP}})$.

Corollaire 4 *(Lawler) Si C est une matrice de distribution alors la permutation identique est la solution optimale du problème de transport (assigment).*

Démonstration : Soit ϕ la solution du problème du transport. Supposons que ϕ n'est pas la permutation identique, alors il existe i, i' tel que $i < i'$ et $\phi(i') < \phi(i)$ alors d'après le Lemme 2 la solution ϕ' de ϕ définie comme suit $\phi'(i) = \phi(i')$ et $\phi'(i') = \phi(i)$ sinon $\phi'(k) = \phi(k)$ est de coût inférieur que celui de ϕ. ∎

Corollaire 5 *Si C est une matrice de distribution alors*

$$\mathbb{E}(L_{T_{PVCP}}) \geq p(1-(1-p)^{n-1}) \sum_{i=1}^{n} c_{ii}$$

Théorème 10 *(Lawler) Si C est une matrice de distribution alors il existe un tour optimal pyramidal.*

Nous allons chercher des conditions pour que le tour optimal (au sens $PVCP$) soit pyramidal.

Pour le PVC l'interaction de chaque sommet est local : chacun est en liaison avec son successeur et son prédécesseur dans un tour donné, cette propriété on peut donner une transformation qui nous permet de construire un tour optimal (au sens PVC) pyramidal pour une matrice C de distribution. Malheureusement pour le PVCP l'interaction est total, donc les conditions d_{ij} positifs ne sont pas suffisant pour déduire l'existence d'un PVPC pyramidal.

On se restreindra ainsi dans la suite au cas particulier du $PVCP$ où tous les sommets sont toujours présents sauf un sommet (on se ramène ainsi à des interactions locales), on notera ce problème $PVCP_{P_0P_1}$ et W le nombre de points présents (qui est une v.a). D'après [34] on a

$$\mathbb{E}(L_T) = \alpha_0 L_{T^0} + \alpha_1 L_{T^1} \tag{2.30}$$

avec

$$\alpha_0 = \sum_{j=0}^{1} \binom{n-2}{j} / \binom{n}{j} Pr(W = n - j)$$

$$\alpha_1 = \frac{Pr(W = n-1)}{n}$$

On note $\max(C,n) = \max_j \{c_{jl} - c_{jk} \text{ pour tout } k,l\}$ et $d_{min} = \min\{d_{ij}\}$

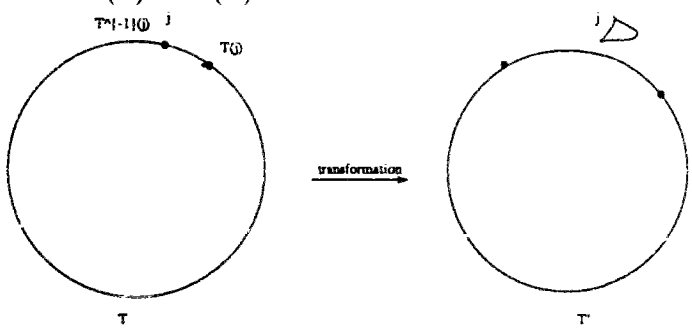
Proposition 17 *Si C est une matrice de distribution alors le $PVCP_{P_0P_1}$ est pyramidal si*

$$3\alpha_1 \max(C,n) \leq \alpha_0 d_{min} \tag{2.31}$$

Démonstration : Nous allons montrer le résultat par récurrence sur le nombre de sommets.

Le résultat est vrai pour $n = 2$, supposons que le résultat est vrai à l'ordre $n-1$. Supposons qu'il reste vrai jusqu'à l'ordre n.

Soit T le tour optimal à travers les n sommets, si T n'est pas pyramidal alors il existe $j \neq n$ tel que $T^{-1}(j) < j > T(j)$. On considère la transformation T' définie par $T'(T^{-1}(j)) = T(j)$, $T'(j) = j$ sinon $T'(k) = T(k)$.



On a donc $c_{jT'(j)} = c_{jj}$ et

$$\mathbb{E}(L_{T'}) = \alpha_0 L_{T'} + \alpha_1 L_{T^n}$$

Donc

$$\mathbb{E}(L_{T'}) - \mathbb{E}(L_T) = \alpha_0(L_{T'} - L_T) + \alpha_1(L_{T^n} - L_{T^1})$$

Or

$$L_{T'} - L_T = c_{T^{-1}(j)T(j)} + c_{jj} - c_{T^{-1}(j)j} - c_{jT(j)}$$

D'après l'équation (2.29) on a

$$\alpha_0(L_{T'} - L_T) = \alpha_0 \sum_{T^{-1}(j)+1}^{j} \sum_{T(j)+1}^{j} d_{kl} \geq \alpha_0 d_{min}$$

De même

$$\begin{aligned} L_{T''} - L_{T'} &= [c_{jj} + c_{T^{-1}(j)T(T(j))} + c_{T^{-1}(T^{-1}(j))T(j)}] \\ &- [c_{jT(T(j))} + c_{T^{-1}(j)T(j)} + c_{T^{-1}(T^{-1}(j))j}] \end{aligned}$$

Donc

$$L_{T''} - L_{T'} \leq 3\max(C, n)$$

D'après (2.31)

$$\alpha_1(L_{T''} - L_{T''}) \leq \alpha_0 d_{min} \leq \alpha_0(L_{T'} - L_T)$$

D'où

$$\mathbb{E}(L_{T'}) \leq \mathbb{E}(L_T)$$

On applique l'hypothèse de récurrence sur le tour $T'\backslash\{j\}$ (une sous-matrice de distribution est une matrice de distribution et $\max\{C, n-1\} \leq \max\{C, n\}$), on a donc un tour pyramidal optimal, par abus de notation T' est le nouveau tour.
Il existe i_j tel que $i_j < j < T'(i_j)$, on considère la transformation T'' de T' : $T''(i_j) = j$, $T''(j) = T'(i_j)$ sinon $T''(k) = T'(k)$. On montre de la même façon que :

$$\mathbb{E}(L_{T''}) \leq \mathbb{E}(L_{T'}) \leq \mathbb{E}(L_T)$$

∎

Heuristique en pire cas constant

Nous allons donner des conditions sur la probabilité pour lesquelles on a une heuristique polynomiale qui est au pire cas constante. On se restreindra dans la suite au cas particulier du $PVCP_{P_0P_1}$ où tous les sommets sont toujours présents sauf un sommet. On rappelle l'heuristique de Christophides [49] :

- Trouver l'arbre de recouvrement minimum $0(n^2)$
- Construire un matching minimum entre les sommets de degré impair $0(n^3)$
- Convertir le tour eulerien en un tour hamiltonien.

On note C le tour construit par cet algorithme, on a d'après [49] :

$$L_{C^0} \leq \frac{3}{2} L_{T^0_{PVC}}$$

Nous allons montrer qu'on a un résultat presque identique dans le cas du $PVCP_{P_0P_1}$.

Proposition 18 *Si la matrice coût est positive et vérifie l'inégalité triangulaire et* $Pr(W \leq n-2) = 0$ *et* $Pr(W = n-1) > 0$ *alors l'heuristique de Christophides est une heuristique au pire cas constante pour le* $PVCP$.

Démonstration :
Comme $Pr(W \leq n-2) = 0$ et $Pr(W = n-1) > 0$ alors $\forall\ T$ on a :

$$\mathbb{E}(L_T) = \alpha_0 L_T + \alpha_1 L_{T^1}$$

La matrice coût étant positive, et T^0_{PVC} étant le tour optimal :

$$\alpha_0 L_{T^0_{PVC}} \leq \alpha_0 L_{T^0_{PVCP}} \leq \mathbb{E}(L_{T_{PVCP}})$$

D'après l'inégalité triangulaire $L_{T^1} \leq 2L_T$, donc

$$\mathbb{E}(L_C) \leq (\alpha_0 + 2\alpha_1)L_{C^0} \leq (\alpha_0 + 2\alpha_1)\frac{3}{2}L_{T^0_{PVC}}$$

$$\mathbb{E}(L_C) \leq \frac{3}{2}(1 + 2\frac{\alpha_1}{\alpha_0})\mathbb{E}(L_{T_{PVCP}})$$

D'après l'équation (2.30) on a

$$\frac{\alpha_1}{\alpha_0} \leq \frac{1}{n-2}$$

On déduit :

$$\mathbb{E}(L_C) \leq \frac{3}{2}(1 + \frac{2}{n-2})\mathbb{E}(L_{T_{PVCP}})$$

∎

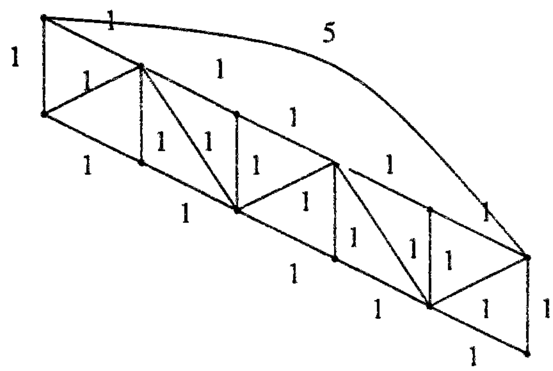
Remarques :

- On remarque que quand n est grand l'heuristique de Christophides pour le *PVCP* se comporte comme dans le cas déterministe (PVC) :

$$\lim_{n\to\infty} \frac{\mathbb{E}(L_C)}{\mathbb{E}(L_{T_{PVCP}})} = \frac{3}{2}$$

Cette borne est la meilleur constante possible, en effet il existe un exemple où la constante $\frac{3}{2}$ est atteinte.

Soit l'exemple proposé par *Cornejols* et *Nemhauser* [19] :



Cornejols et *Nemhauser* montrent que la borne est atteinte pour ce problème :

$$\lim_{n\to\infty} \frac{L_{C^0}}{L_{T^0_{PVC}}} = \frac{3}{2}$$

Or on est dans le cas où les villes sont placées sur leur enveloppe convexe donc d'après la Proposition 2 on a $PVCP \equiv PVC$ et $\frac{L_{C^0}}{L_{T^0_{PVC}}}$ est bornée d'ou le résultat.

- La Proposition 18 se généralise au cas où tous les sommets sont toujours présents sauf k sommets et k indépendant de n.

$$Pr(W \leq n-k-1) = 0 \text{ et } Pr(W = n-k) > 0$$

Proposition 19 *Si la matrice coût est positive et vérifie l'inégalité triangulaire et* $Pr(W \leq n-k-1) = 0$ *et* $Pr(W = n-k) > 0$ *alors l'heuristique de Christophides est une heuristique au pire cas constante pour le* $PVCP$.

$$I\!E(L_C) \leq \frac{3}{2}[1 + \frac{k^2(k+1)}{n-2}]I\!E(L_{T_{PVCP}}).$$

Démonstration :

D'après [34] on a

$$\mathbb{E}(L_T) = \sum_{r=0}^{k} \alpha_r L_{T^r}$$

$$\alpha_r = \sum_{j=r}^{k} \binom{n-2-r}{j-r} / \binom{n}{j} Pr(W = n-j) \text{ pour tout } r \in [0..k] \tag{2.32}$$

D'après l'inégalité triangulaire on a :

$$L_{T^r} \leq (r+1)L_{T^0} \quad 1 \leq r \leq k$$

D'après [38] on a :

$$\alpha_r \leq \alpha_{r'} \quad 1 \leq r' \leq r \leq k$$

Donc

$$\frac{\alpha_r}{\alpha_0} \leq \frac{\alpha_1}{\alpha_0} \quad 2 \leq r \leq k$$

Du fait que $l\binom{n}{l} = n\binom{n-1}{l-1}$, on a d'après (2.32)

$$(n-2)\alpha_1 = (n-2)\sum_{j=1}^{k} \binom{n-2-1}{j-1} / \binom{n}{j} Pr(W = n-j) \leq k\alpha_0$$

$$\frac{\alpha_1}{\alpha_0} \leq \frac{k}{n-2}$$

On a donc

$$\mathbb{E}(L_C) \leq \frac{3}{2}[1 + \frac{k^2(k+1)}{n-2}]\mathbb{E}(L_{T_{PVCP}}).$$

∎

2.4 Implantation des Algorithmes

Le PVCP étant $NP - complet$, on doit donc faire appel à des heuristiques pour résoudre le problème. Nous nous intéresserons en premier lieu à des heuristiques d'optimisation locales généralisées telle que le recuit simulé et l'algorithme tabou, on discute surtout des possibilités d'accélérer ces algorithmes. Puis nous traitons une autre catégorie d'heuristique (heuristique conçue pour le problème de très grande taille), l'algorithme de partitionnement de Karp. Ces algorithmes sont testés pour des voisinages et pour des approximations différentes.

La partie suivante est consacrée à une étude détaillée de l'approche du recuit simulé. En particulier, on expliquera l'analogie avec la mécanique statistique et on présentera les détails de notre implémentation en décrivant le paramètrage et les procédures appelées. Des résultats numériques seront données au fur et à mesure de cette partie.

2.4.1 Recuit Simulé

Le recuit simulé (RS) est une nouvelle approche pour approximer les solutions des problèmes d'optimisation combinatoires. Cette méthode s'est révélée efficace pour traiter les POCs difficiles [43].

Le recuit simulé est une version probabiliste de l'algorithme d'optimisation locale (l'optimisation locale est un algorithme itératif de recherche de minima dans un voisinage d'une solution initiale donnée). Dans un schéma de recuit, la transition d'une solution vers une solution voisine est aléatoire ce qui permet de retenir éventuellement des solutions de coût supérieur et d'éviter ainsi de stagner dans un minimum local (par opposition à l'optimisation locale où on transite toujours vers les solutions de moindre coût).

Les excellents résultats obtenus pour le recuit simulé ont incité un grand nombre de mathématiciens et de physiciens à s'intéresser à la méthode (Voir Laarhaven and Aarts [1]). Johnson [43] ont étudié les performances du recuit simulé pour quatre problèmes d'optimisation, le PVC (Problème du Voyageur de Commerce), le PPG (Problème de Partitionnement des Graphes), le PPN (Problème de Partition des Nombres) et le PCG(Problème de Coloration des Graphes). Ils ont remarqué que le recuit est performant pour le PVC et le PGP dans le sens qu'il donne une bonne solution en un temps raisonnable, ce qui n'est pas le cas pour les deux autres problèmes. En effet pour le PPN on obtient une solution qui est déviée considérablement du minimum global alors que pour le PCG on a une bonne solution mais avec un temps d'exécution énorme. Morgenster et Shapiro [56] reprennent le recuit simulé pour le PCG en utilisant un mécanisme de transition différent de celui utilisé par Johnson et ils trouvent une même qualité de solution mais pour un temps d'exécution raisonnable (pour un graphe de 1000 sommets le temps est réduit de $22h$ à $4h5$'). Ainsi le comportement du recuit pour les POCs reste encore mal compris. Dans la suite, nous allons étudier les performances de cette approche pour le PVCP.

Concept du recuit simulé

On rappelle qu'un algorithme d'optimisation locale consiste à explorer partiellement l'ensemble des solutions possibles de la façon suivante : on part d'une solution initiale aléatoire puis on applique une transformation afin d'obtenir une nouvelle solution. Si le coût de cette nouvelle solution est inférieur au coût de la solution courante on oublie la première solution

et on retient la nouvelle. On répète l'opération jusqu'à ce qu'aucune transformation dans le voisinage ne produise de solution meilleure. Généralement cette procédure ne conduit pas au coût minimum car elle finit piégée dans un minimum local. Kirkpatrick, Gelatt et Vecchi [47] ont introduit les algorithmes de recuit. Dans cette approche, la transition d'une solution vers une solution voisine se fait à coup sûr si la différence de coût qu'on note Δ est négative et sinon avec une probabilité $exp(-\Delta/T)$, où T est la température qui représente un paramètre de contrôle de l'amplitude des mouvements faisant sortir des minima. On remarque que quand la température est nulle, l'algorithme devient déterministe et l'on retrouve un algorithme d'optimisation locale. En revanche quand la température n'est pas nulle les solutions de coût inférieur sont favorisées. Aux basses températures, les probabilités d'observer des solutions de faible coût sont grandes par rapport aux solutions de coût élevé. Plus la température augmente, plus les solutions de coût élevé interviennent. Pour déterminer les minima de la fonctionnelle associée à un POC donné, on introduit une double dynamique :

- On cherche le minimum de la fonctionnelle à T fixé.
- On diminue progressivement T.

On remarque que si on prend une température initiale infinie, on obtient un algorithme d'énumération complète.

1- Prendre une solution initiale S

2- Tant qu'il y a un voisin S' de S non testé

2-1 Calcul de Δ
Δ = coût(S')-coût(S)

2-2 si $\Delta < 0$ alors
$S \equiv S'$

Optimisation locale

1- Prendre une solution initiale S

2- Tant que la température minimale n'est pas atteinte

2-1 Tant que l'équilibre n'est pas atteint

2-1-1 générer un voisin aléatoire S' de S

2-1-2 Calcul de Δ
Δ = coût(S')-coût(S)

2-1-3 si $\Delta < 0$ alors $S \equiv S'$

sinon $S \equiv S'$
avec la probabilite $exp(-\Delta/T)$

2-2 $T = \alpha T$

Recuit Simulé

Liens avec la mécanique statistique

Le recuit physique consiste a baisser progressivement la température du système pour l'amener à un état stable (énergie minimale) sans le piéger dans un état métastable (minimum local). Cette méthode fut d'abord mise en œuvre sur les problèmes de "verres de spin" en physique de la matière désordonnée. Puis on a remarqué que les POC's sont caractérisés par des contraintes antagonistes qui les rapprochent des systèmes frustrés désordonnés tels que les verres de spin. En se fondant sur cette analogie on transpose la méthode aux POCs, on aura ainsi les correspondances suivantes :

Etat du système	$\Longleftrightarrow$	Solution d'un POC
Energie d'un état	$\Longleftrightarrow$	Fonction coût de la solution
minimum d'énergie	$\Longleftrightarrow$	Solution optimale

Résultats mathématiques

Lundy et Mees [53] donnent des justifications mathématiques pour cette approche et montrent que l'analyse de cet algorithme se ramène à l'étude d'une suite de chaînes de Markov homogènes (à température fixée, on a une chaîne de Markov homogène). Ils montrent que sous certaines conditions (refroidissement lent i.e T décroît en $1/lnT$ et chaîne irréductible i.e on peut atteindre la solution optimale à partir de n'importe quelle solution initiale), la suite de chaînes de Markov générée par l'algorithme du recuit converge vers une loi limite qui charge exclusivement l'ensemble des solutions optimales. Malheureusement Sasaki et Hajek [59] ont montré que le nombre de transitions nécessaires pour avoir la convergence (pour la plus part des problèmes) est plus grand que la taille de l'ensemble des solutions (pour le PVC à travers n sommets la vitesse de convergence est de l'ordre de $O(n^{n^{2n-1}})$ alors que l'ensemble des solutions est de cardinal $(n-1)!$). Dans la pratique, on se contente d'une version approchée de cet algorithme qui converge en un temps polynomial et donne une solution presqu'optimale pour la plupart des problèmes.

Implantation du Recuit Simulé pour le Problème du Voyageur de Commerce Probabiliste Euclidien

Nous allons préciser les notions de solution, voisinage, solution initiale, puis nous décrirons le programme de refroidissement.

On considère n villes uniformément distribuées dans le carré $[0,1]^2$, et soit $d(i,j)$ la distance euclidienne entre le sommet i et le sommet j. Soit p la probabilité de présence de chaque ville. On note G(n,p) le PVCP associé à n et p. On se propose de chercher un tour qui minimise la fonctionnelle associée au PVCP, donnée par la formule (2.2).

Une solution est une permutation cyclique. On choisit le voisinage deux-inter-change, notion qui a été introduite pour le PVC [10]. Dans cette structure de voisinage, deux tours sont voisins s'ils se déduisent l'un de l'autre par inversion de deux points :

$(1,...i-1,i,i+1,...j-1,j,j+1,....,n) \Longrightarrow I(i,j) = (1....i-1.j.j-1....i+1,i,j+1,...,n)$

On peut calculer récursivement la différence de coût [5]. En fait, on calcule la différence entre $I(i,j)$ et $I(i+1,j-1)$. On note $\Delta_{i,j}$ la différence de coût de $I(i,j)$ et de la solution

optimale, on introduit deux matrices A et B définies comme suit :

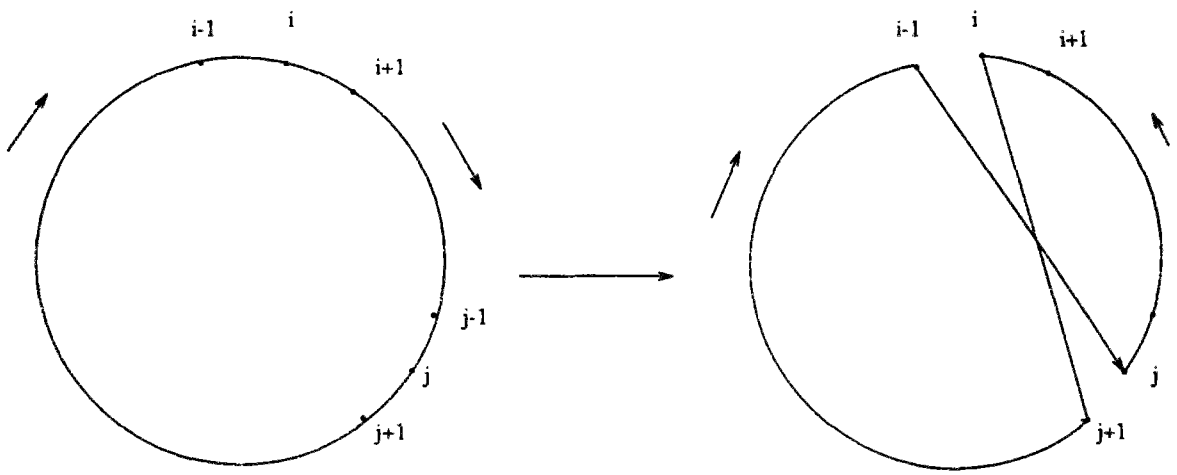
$$A_{ik} = \sum_{r=k}^{n-1} (1-p)^{r-1} d(i, i+r) \tag{2.33}$$

$$B_{ik} = \sum_{r=k}^{n-1} (1-p)^{r-1} d(i, i-r) \tag{2.34}$$

ainsi on aura pour j=i+k

$$\begin{aligned} \Delta_{i,j} &= \Delta_{i+1,j-1} \\ &+ p^2[((1-p)^{-k}-1)A_{ik+1} + ((1-p)^k - 1)(B_{i1} - B_{in-k}) \\ &+ ((1-p)^k - 1)(A_{j1} - A_{jn-k}) + ((1-p)^{-k} - 1)B_{jn-k+1} \\ &+ ((1-p)^{k-n} - 1)A_{jn-k+1} + ((1-p)^{n-k} - 1)(B_{j1} - B_{jn-k}) \\ &+ ((1-p)^{n-k} - 1)(A_{i1} - A_{ik}) + ((1-p)^{k-n} - 1)B_{in-k+1}] \end{aligned} \tag{2.35}$$

En utilisant une implémentation astucieuse des matrices A et B, il est possible de calculer les éléments de ces matrices en $O(n^2)$.



configuration initiale configuration I(i,j)

Programme de Refroidissement

Une implémentation en temps polynomial de l'algorithme du recuit simulé peut être réalisé en générant une chaîne de Markov homogène de longueur finie (nombre de voisins à chaque étape) et une suite décroissante finie de températures (nombre de transitions fini). L'algorithme nécessite la spécification de quatre paramètres qu'on appelle programme de refroidissement.

- Température initiale :

  La valeur initiale de la température T_0 est déterminée de façon que presque toutes les transitions soient acceptées au début.

- Règle de transition de la température

  Dans l'algorithme de décroissance de la température $T_{n+1} = \alpha T_n$, on prend une valeur de α proche de 1 afin que la restauration du quasi-équilibre soit atteinte rapidement.

- Température finale :

  Le critère d'arrêt est défini par une valeur fixe (on prend une valeur pour laquelle on juge que le quasi-équilibre est atteint). Afin de diminuer le temps perdu, on décide aussi de s'arrêter si on a un gel (c'est-à-dire pas d'amélioration du critère après un certain nombre d'itérations).

- Longueur de la chaîne :

  On prend une longueur L de la chaîne de Markov homogène, constante et indépendante de la température.

Résultats expérimentaux

Les algorithmes ont été implémeté sur un ordinateur "SPARC station 1+" . Les résultats sont testés sur des graphes ayant entre 25 et 200 sommets et des probabilités 0.1, 0.5, 0.7, et 0.99.

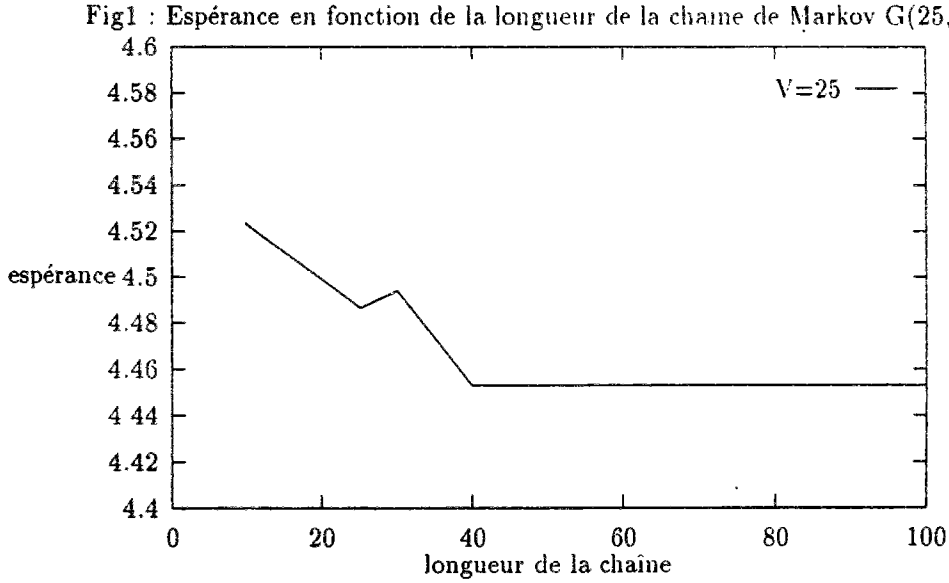
La complexité du recuit dépend surtout du choix du programme de refroidissement et du choix du voisinage. Afin d'obtenir une bonne implémentation de cet algorithme pour notre problème, nous commençons par chercher le programme de refroidissement optimal. Ensuite nous comparons les performances du recuit avec un algorithme d'optimisation locale et d'autres algorithmes déjà existant.

1. Optimisation des paramètres

   Nous allons décrire les résultats numériques qui nous ont permis de déterminer les paramètres optimaux du programme de refroidissement (température finale et longueur de la chaîne...)

   - Influence de la longueur de la chaîne

Fig1 : Espérance en fonction de la longueur de la chaine de Markov G(25,0.99)

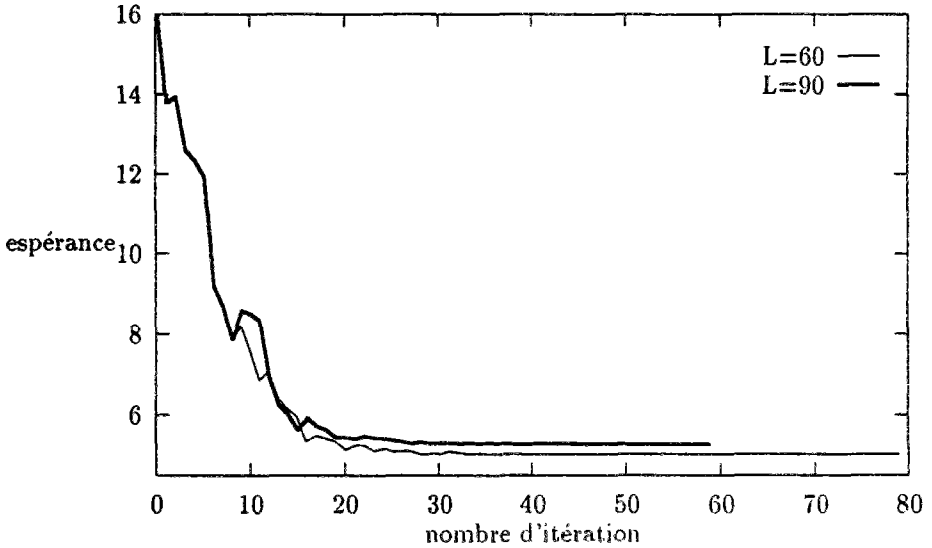


L	Espérance	Temps C.P.U
10	5. 355135	140.35
40	5.222456	708.39
50	5.090686	852.07
70	5.025467	1130.66
90	5.275300	1394.34

Tableau 2.1 : influence de la longueur de chaîne pour un G(50,0.7)

On fixe tous les paramètres du programme de refroidissement sauf la longueur de la chaîne L qu'on fait varier de 20 jusqu'a 100. L a évidemment un effet direct sur la qualité de la solution finale. La Figure-1 montre qu'a partir d'un certain rang de L, on a la même qualité de solution. On sait que la longueur de la chaîne de Markov homogène doit être choisie de façon qu' on puisse visiter le plus grand nombre de solutions dans le voisinage d'une solution donnée. On remarque d'après la Figure-1 qu'il suffit de prendre L de l'ordre de la taille du problème, pour cette valeur le quasi équilibre est atteint. D'après le Tableau 2.1 si L est trop petit la solution finale dévie considérablement de la solution optimale, L trop grand donne un temps d'exécution énorme (supérieur de 20%) sans amélioration notable de la qualité de la solution.

Fig2 : Evolution de la solution durant l'exécution pour différent L G(50,0,7)



Pour les températures élevées il n'est pas intéressant de passer trop de temps car on n'explore pas un domaine important de l'espace des solutions, en effet presque toutes les solutions proposées sont acceptées. D'après l'implémentation de Kirkpatrick, il faut utiliser une autre condition d'arrêt (dans le but de diminuer des tirages non intéressants au début du refroidissement). Si on a un certains nombres de succès alors on change de température. Cette méthode appelée coupe améliore le temps d'exécution sans détériorer la qualité.

Contrairement à ce qu'on pense et d'après la Figure-2, si on augmente trop la longueur de la chaîne, on peut détruire la qualité de la solution. Ceci est dû au

Tempér initiale	T_0 avec coupe	T_0 sans coupe	T_0 - 0.3	T_0 - 0.4	T_0 - 0.6	T_0 - 0.9
Espérance	5.0296	5.2705	5.0432	5.1000	5.1084	5.5794
Temps C.P.U	369.39	428.79	329.59	289.37	212.46	100.77

Tableau 2.2 : Comparaison des deux méthodes pour un G(50,0.7)

L	α				
	0.6634	0.8145	0.9025	0.9500	0.9747
2	5.5895	5.6797	5.2826	5.3841	6.4145
4	5.5244	5.3458	5.4077	5.1986	5.8466
8	5.4422	5.2075	5.3419	5.3953	6.1467
16	5.3128	5.1856	5.1767	5.02962	5.6105
32	5.4407	5.2404	5.0789	5.0455	6.1478
64	5.3371	5.0304	5.1138	5.0418	5.7007
128	5.2617	5.0255	5.25126	5.1113	5.6244

Tableau 2.3 : l'espérance en fonction de la longueur de la chaîne et de α pour un G(50,0.7)

fait qu'on a deux conditions d'arrêt et que la deuxième condition dépend de la première : si on augmente L alors pour plusieurs températures c'est la deuxième condition qui va être la condition d'arrêt.

- Comparaison entre coupe et diminution de la température

  Diminuer la température initiale nous permet d'éviter des tirages non intéressants. D'après le Tableau 2.2 les deux méthodes sont presque équivalentes. En effet prendre une température initiale $T_0 = 1.3$ et coupe= 10 donne presque la même performance que température initiale = 1 et coupe= 0. Par la méthode de coupe on a une légère détérioration de la solution de 5% mais un temps d'exécution meilleur (moindre de 10%). On choisira dans la suite la méthode de coupe et L de l'ordre de la taille du problème.

- Température finale

  On décide de s'arrêter si on a un gel c'est a dire pas d'amélioration après cinq itérations successives de l'algorithme de décroissance de la température. Cette méthode donne de meilleurs résultats que si on gèle le recuit lorsque le taux d'acceptation (nombre de transitions acceptées divisé par le nombre de transitions proposées) est inférieur à un seuil donné suffisamment petit (en général de l'ordre de 2%).

- Influence de la température

  - Incrémentation de la température

    Les paramètres L et α contrôlent le temps nécessaire pour la convergence du recuit. Les tableaux 2.3 et 2.4 illustrent le comportement du recuit en fonction de ces facteurs. On fixe tous les paramètres sauf ces deux derniers, puis on fait varier α sur l'intervalle [0.6..0.98] et L sur [1..120].

L	α				
	0.6634	0.8145	0.9025	0.9500	0.9747
2	22.23	30.61	40.76	54.12	72.50
4	34.49	40.43	58.90	94.52	134.65
8	51.56	70.99	123.45	190.01	134.65
16	87.68	131.25	199.41	369.39	521.07
32	147.03	234.87	419.41	617.73	870.28
64	205.04	424.86	769.39	970.67	1246.88
128	385.20	664.97	1221.46	1478.96	1584.35

Tableau 2.4 : temps d'exécution en fonction de L et de α pour un G(50,0.7)

Le tableau 2.3 donne la valeur de notre fonctionnelle pour chaque combinaison des paramètres. On remarque que la qualité de la solution est généralement meilleure quand on augmente le temps d'exécution. Prendre une valeur de α proche de 1 et donc un algorithme de décroissance de T assez lent permet d'améliorer les résultats du recuit. Dans ce cas, on a une assez bonne solution car le quasi équilibre est rapidement atteint mais pour atteindre le gel on risque d'itérer plus longtemps.

On prendra dans la suite comme paramètre standard α= 0.9500. Pour cette valeur, la suite (indexée par T) des distributions d'équilibres des chaînes de Markov homogènes (à T fixée) est "quasi-stationnaire" et un petit nombre de transition est généralement suffisant pour rétablir le quasi équilibre [43] .

- Température initiale

  Le choix de la température initiale a un effet considérable sur la qualité de la solution finale. En effet si on prend une température initiale trop grande on peut avoir un algorithme d'énumération complète et un temps d'exécution infini. De même pour une température initiale nulle on a un algorithme d'optimisation locale et on peut donc obtenir une très mauvaise solution. On se propose de chercher une bonne initialisation de la température. Les méthodes généralement utilisées consistent à déterminer la température initiale de façon que le rapport d'acceptation (nombre de transitions acceptées divisé par le nombre de transitions proposées) soit proche de 1. Nous allons comparer la méthode de Aart [1] et celle proposée par Chervi [15].

  Méthode de Aart.

  On prend une température initiale petite et on la multiplie par une constante plus grande que 1 et on répète cette opération jusqu'a ce que la valeur du rapport d'acceptation χ_0 calculé d'après les transitions générées soit proche de 1. En physique cette méthode correspond à chauffer le solide jusqu'à ce que les particules soient dispersées aléatoirement.

  Méthode de Chervi

  la méthode consiste à calculer la moyenne des différences de coût positives qu'on note ΔC^+ pour un nombre de transitions aléatoire et choisir un rapport

	Chervi	Aart
espérance	3.80547	3.80547
Temps C.P.U	70.53	82.12

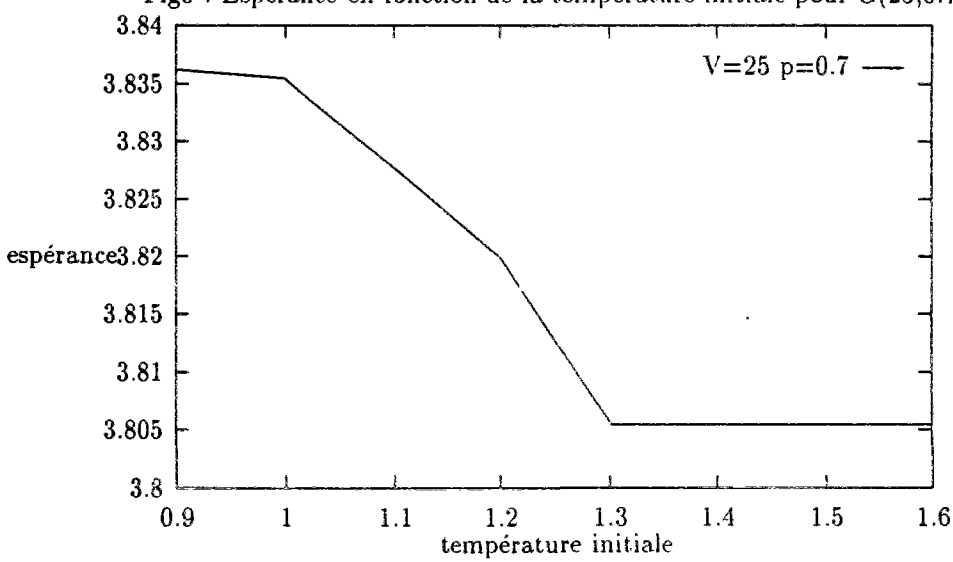
Tableau 2.5 : comparaison des deux méthodes pour un G(25,0.7)

d'acceptation χ_0 proche de 1 et on calcule donc T_0 d'après la formule suivante.

$$T_0 = \Delta C^+ / ln(\chi_0) \tag{2.36}$$

On remarque d'après le Tableau 2.5 que les deux méthodes permettent d'obtenir la même solution mais avec un meilleur temps d'exécution pour la méthode de Chervi. En effet pour la méthode de Aart, la température initiale est assez grande et donc une amélioration de la solution n'apparaît qu'après un grand nombre de tirages.
Dans la Figure-3, nous représentons les valeurs obtenues de la fonctionnelle associée au PVCP, pour différentes valeurs de la température initiale dans la méthode de Aart. Cette méthode donne la valeur critique $T_0 = 1.3$ pour le PVCP puisque si on diminue encore la température initiale on aboutit à une mauvaise solution sans amélioration du temps d'exécution. Dans la suite des simulations, nous prendrons $T_0 = 1.3$.

Fig3 : Espérance en fonction de la température initiale pour G(25,0.7)



2. Influence de la fonctionnelle sur le rapport d'acceptation

Dans cette section, nous étudions le comportement du rapport d'acceptation pour la fonctionnelle du PVCP en fonction de p, probabilité de présence de chaque ville. Nous comparons notamment le taux d'acceptation pour le PVCP, pour p proche de 1 (p = 0.99), avec le PVC (qui correspond au PVCP pour p = 1).

Pour une bonne implémentation du recuit il faut que la différence de coût des solutions voisines soit petite. En effet quand le coût de la nouvelle solution obtenue est très

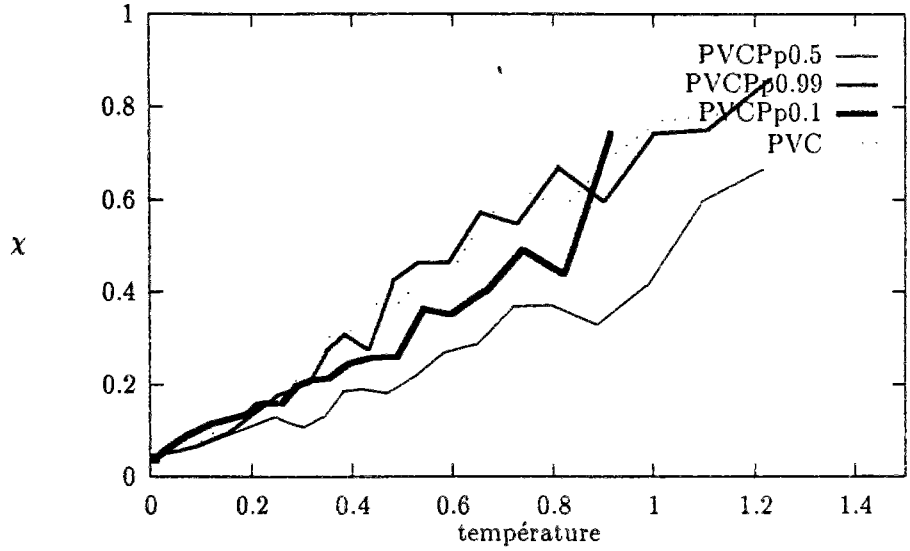
supérieur au coût de la solution courante, la probabilité d'accepter la nouvelle solution ($exp(-\Delta/T)$) est faible. En conséquence, quand le voisinage d'un tour contient un grand nombre de tours de coûts beaucoup plus grands, le nombre de transitions non retenues est d'autant plus important que la température est faible. Une transition refusée étant synonyme de temps perdu, il faut donc utiliser des transformations pour lesquelles le voisinage de toute configuration ne contient que des configurations de coûts peu différents. Pour le PVC, la différence de coût Δ après une transition ne peut pas dépasser $2(\max_{i \neq j}(d(i,j)) - \min_{i \neq j}(d(i,j))$. Pour le PVCP et d'après le Tableau 2.6, Δ est généralement plus petit que pour le PVC.

La différence de coût pour le PVCP dépendant de p, la probabilité de transition et donc le rapport d'acceptation varient en fonction de p. Dans la Figure-4, nous représentons l'évolution du taux d'acceptation pour le PVCP (pour différentes valeurs de p) et pour le PVC. Cette Figure illustre le comportement attendu c'est-à-dire que pour des températures élevées, presque toutes les transitions proposées sont acceptées et le taux d'acceptation est proche de 1. Plus la température décroît, moins les transitions sont acceptées et ainsi pour les températures très petites, aucune transition n'est acceptée. On remarque aussi que le rapport d'acceptation est presque le même pour le PVCP et le PVC quand p est proche de 1, par contre il diminue quand p décroît jusqu'a 0.5 et puis croît pour $p < 0.5$. La valeur $p = 0.5$ paraît être la probabilité critique (notre transition est de moins en moins adaptées quand p décroît jusqu'a 0.5). pour la température $T = 0.9$ on a plus de 40% de solution rejetées pour $p = 0.5$ que pour les autres probabilité. D'après la Figure-4 on peut penser que la différence de coût pour le PVCP croît jusqu'a $0 \leq p \leq 0.5$ et décroît pour $0.5 \leq p \leq 1$: du fait que le nombre de transition proposées ne dépend que de n la taille du problème, donc la décroissance du rapport d'acceptation à n fixé se traduit par une décroissance du nombre de transition acceptées, ainsi en supposant que le nombre de solution de coût plus petit que la solution courante est du même ordre pour toutes les probabilités alors le nombre de transition acceptées est équivalent au nombre de solution voisine de coût supérieur à la solution courante. Donc plus que la différence de coût est grande plus que la probabilité d'accepter la solution est faible.

température	T_0	T_{10}	T_{20}	T_{50}	T_{60}	T_{80}	T_{90}	T_{100}
χ_{PVCP}	0.7962	0.3342	0.3172	0.2291	0.2206	0.1566	0.1201	0.1162
χ_{PVC}	0.7992	0.3754	0.1181	0.0572	0.0464	0.0468	0.0436	0.0460

Tableau 2.6 : Comparaison des temps perdus en transitions rejetées pour le PVC et PVCP où p=0.99

Fig4 : Rapport d'acceptation en fonction de la température pour un G(25,p)



on rappelle que χ est le rapport d'acceptation = nombre de transitions acceptées / nombre de transitions proposées

3. Complexité

Les temps d'exécution donnés dans le Tableau 2.7 incluent le temps pour générer le graphe, initialiser le tour et la température mais cette période initiale ne nécessite pas plus de 2% du temps total et est donc négligeable. Par contre 80 % du temps est consacré au calcul des différences de coût ce qui est énorme.

En comparaison avec le PVC, l'algorithme du recuit pour le PVCP nécessite à qualités de solutions comparables, un temps d'exécution beaucoup plus important (voir Tableau 2.8). Ceci est dû à deux inconvénients liés directement au PVCP qui sont, la topologie et le caractère des interactions.

- Topologie

  C'est surtout le choix du voisinage qui fixe l'efficacité du recuit simulé pour les POCs. Dans notre implémentation, nous avons utilisé deux-inter-change comme voisinage. Cette topologie donne de bons résultats pour le PVC (voir [13] et [10]) mais s'avère coûteuse pour le PVCP (voir Tableau 2.7 et Tableau 2.8). En effet, pour le PVC le calcul de la différence de coût se fait en un temps constant indépendamment du nombre de villes. Cette différence est donnée par la formule suivante (si deux-inter-change est effectué sur la chaîne $(i, \ldots\ldots, j)$).

$$\Delta = d(i-1,j) + d(i,j+1) - d(i-1,i) - d(j,j+1) \qquad (2.37)$$

V	p	Espérance finale	Temps C.P.U
100	0.99	8.457719	1394.33
	0.7	7.611748	1290.07
	0.5	6.614403	3410.32
	0.1	3.102383	2643.94
200	0.99	11.04591	2869.37
	0.7	10.11375	2967.68
	0.5	9.38152	5876.11
	0.1	4.202085	2760.98

Tableau 2.7 : Complexité du recuit pour le PVCP

Par contre pour le PVCP la différence de coût se calcule d'après (2.35) en un temps de l'ordre de $0(n^2)$ tout en gardant en mémoire les deux matrices A et B, et donc un encombrement mémoire de l'ordre de $0(n^2)$. La complexité de notre implémentation est de l'ordre de $0(\tau LK)$ où L est la longueur de la chaîne de Markov homogène, K le nombre d'incrémentations de la température et τ est le temps d'exécution d'une seule transition. Donc un bon voisinage doit aussi impliquer un calcul de différence de coût simple et rapide. Cette propriété n'est pas vérifié par le PVCP (τ est en $0(n^2)$) ce qui explique le temps d'exécution énorme qu'on obtient (voir Tableaus 2.7 et 2.8).

- Interaction

  L'algorithme du recuit pour les POCs se base sur une analogie avec le comportement des verres de spin. Or pour les verres de spin l'interaction de chaque site (qui correspond à une ville pour notre problème) est locale, dans le sens que chaque site est en relation avec un petit nombre de sites voisins. Cette propriété reste vraie pour la plupart des POCs (par exemple pour le PVC chaque ville est en interaction avec son successeur et son prédécesseur) mais elle n'est pas vérifiée pour notre problème. En effet, pour le PVCP chaque sommet est en interaction avec tous les autres (voir 2.35) et on a donc une interaction totale. Une transition d'une solution vers une solution voisine implique une très grande perturbation de la solution courante (si on est déjà dans une assez bonne solution, de telles perturbations peuvent générer une mauvaise solution). D'autre part et à cause du caractère global des interactions, le calcul de la différence de coût se fait au meilleur des cas en n opérations et donc même si on change de topologie on aura toujours un temps d'exécution considérable.

4. Performances du recuit

   Pour étudier les performances du recuit simulé pour le PVCP, nous commençons par le comparer à un algorithme d'optimisation locale. Les valeurs du critère associé au PVCP déterminés par les deux algorithmes (Tableau 2.9) sont comparables. En effet le pire des cas trouvés pour le recuit simulé est plus petit de 1% que le meilleur tour trouvé par optimisation locale, mais le temps d'exécution du recuit est beaucoup plus grand.

V	PVCP $p = 0.99$	PVC
25	71.72	6.80
50	420.33	21.41

Tableau 2.8 : Comparaison des Temps d'exécution du recuit simule pour le PVC et PVCP

nb d'exécut	p	recuit simule k exécution	opt locale k exécut
1	0.7	3.804305	3.820076
	0.1	1.153049	1.152749
10	0.7	3.820030	3.820076
	0.1	1.152714	1.152749
25	0.7	3.804305	3.820076
	0.1	1.152714	1.152749

Tableau 2.9 : comparaison du recuit simulé avec optimisation locale

Ainsi, une exécution du recuit nécessite un temps de calcul de l'ordre de 75 exécutions de l'algorithme d'optimisation locale, ce qui correspond au nombre de transitions de la température.

D'après les travaux de Chevri [15], l'heuristique de tri radial (les villes sont triées suivant l'angle radial avec leurs barycentre) combiné avec deux-inter-change paraît être l'algorithme le plus adapté pour le PVCP pour p petit, dans le sens qu'il donne une bonne solution rapidement. Mis à part le temps d'exécution, le recuit simulé donne un tour qui est compétitif avec les heuristiques conçues pour le PVCP (voir Tableau 2.10). Cependant, le recuit devient impraticable pour un grand nombre de sommets (Tableau 2.7).

Une comparaison avec l'algorithme Tabou sera donnée ultérieurement.

V	p	recuit simule	Tri radiale+2-int d'après [15]
25	0.7	3.804305	3.625295
	0.1	1.153049	1.038632
50	0.7	5.025467	5.226217
	0.1	2.097089	1.911436

Tableau 2.10 : Comparaison du recuit simulé et de la meilleure heuristique trouvée pour le PVCP

2.4.2 Changement de Topologie

La vitesse de convergence du recuit varie considérablement en fonction de la topologie ([56]). Par exemple pour le PCG, Morgenster et Shapiro [56] reprennent le recuit simulé en utilisant un mécanisme de transition différent de celui de Jhonson. Ils obtiennent une même qualité de solution et avec un temps d'exécution raisonnable.

Nous introduisons dans ce paragraphe de nouveaux mécanismes de transition, puis nous comparons les résultats respectifs.

Structure de cycle

On sait que si le nombre de villes n est un nombre premier, les T^r définis dans (2.3) sont des tours. On rappelle que la structure de cycle pour un tour T qu'on note C_T est définie par :

$$C_T = (T, T^1, T^2, ..., T^{k-1}, T^{k-1}, ..., T) \tag{2.38}$$

où $k = (n-1)/2$.

Pour n premier, Les structures de cycles forment une partition de l'ensemble des solutions en classes d'équivalence et les espérances des tours T^r se déduisent directement de la structure de cycle d'après (2.18). Pour parcourir l'ensemble des solutions, nous appliquons une procédure d'optimisation à l'intérieur d'une structure de cycle, puis pour générer de nouvelles solutions, nous utilisons deux-inter-change et nous continuons l'investigation à l'intérieur de la nouvelle structure de cycle. Une structure de cycle contient $n/2$ tours différents, le nombre d'itérations (à l'intérieur d'une structure) est donc limité Pour ce voisinage le calcul de la différence de coût est de l'ordre de $0(n)$: si un tour T^i est choisi dans C_T et si les L_{T^r} sont stockés alors

$$\Delta_{struct} = \sum_{r=0}^{k-1} [q^r - q^{i(r+1)+r} + q^{n-2}(q^{-r} - q^{-(i(r+1)+r)})] L_{T^r}. \tag{2.39}$$

Shift

Une autre topologie est proposée par Bertsimas and Howell [8]. elle consiste à déplacer un seul sommet.

Par exemple, soit le tour initial

$$(1, ...i-1, i, i+1, ...j-1, j, j+1, ..., n)$$

alors shift de i après j donne le tour suivant :

$$I(i,j) = (1, ..., i-1, i+1, ..., j-1, j, i, j+1, ..., n)$$

Pour cette topologie, la différence de coût peut être calculée récursivement (comme pour deux-inter-change, on calcule la différence entre $I(i,j)$ et $I(i,j-1)$voir [5]). On note $\widehat{\Delta}_{ij}$ le coût de $I(i,j)$.

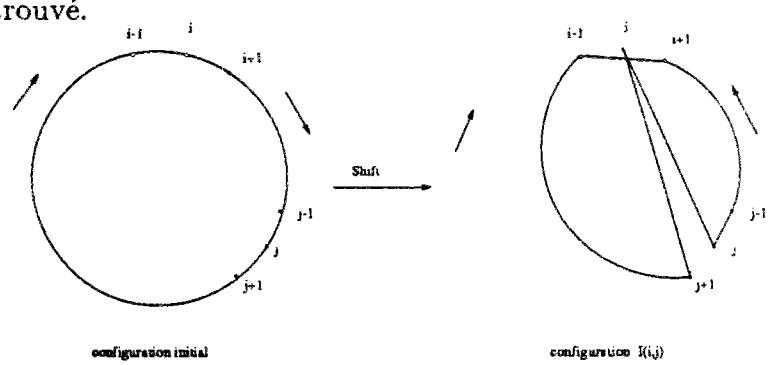
$$\begin{aligned}
\widehat{\Delta}_{ij} &= \widehat{\Delta}_{ipredc[j]} \\
&+ p^2[((1-p)^{-k} - (1-p)^{-(k-1)})A_{ik+1} + ((1-p)^k - (1-p)^{k-1})(B_{i1} - B_{in-k}) \\
&+ p(A_{jn-k} - A_{j1}) + ((1-p)^{-1} - 1)B_{jk+1} \\
&+ (p(1-p)^{n-k-1})(A_{i1} - A_{ik}) - p((1-p)^{k+n})B_{in-k+1} \\
&+ ((1-p)^{-1} - 1).A_{jn-k+1} + p(B_{j1} - B_{jn-k})]
\end{aligned} \tag{2.40}$$

		RS + Structurecycle		*RS + Shift*		*RS + deuxchange*	
V	*p*	espérance	Temps C.P.U	Espérance	Temps C.P.U	Espérance	Temps C.P.U
23	0.7	3.77144	28.81	3.73484	80.42	3.73484	74.86
	0.1	1.04182	48.28	1.04148	73.83	1.04163	63.02
47	0.7	4.86102	166.03	4.89550	507.23	4.89550	375.72
	0.1	1.99932	235.73	1.99003	609.26	1.99054	349.76

Tableau 2.11 : comparaison des topologies

Comparaison des différentes topologies

La topologie structure de cycle est mieux adaptée que deux-inter-change ou shift. En effet, nous obtenons une même qualité de solution avec un temps d'exécution plus réduit (Tableau 2.11). Ce résultat est prévisible car le calcul de la différence de coût pour la structure de cycle se fait en un temps de l'ordre de $0(n/2)$. Malheureusement cette topologie n'est valable que lorsqu'on a un nombre premier de villes. Pour un nombre non premier, nous proposons la stratégie suivante : on cherche le plus petit nombre premier p_1 supérieur ou égal à n, on ajoute $n - p_1$ point, puis on cherche le tour à travers les p_1 points par l'algorithme du recuit simulé muni de la topologie structure de cycle, ensuite on "gomme" les points qu'on a rajouté du tour trouvé.



2.4.3 Méthode Tabou

Nous donnerons tout d'abord une description générale de la méthode tabou (Tabou Search TS). Puis nous présenterons l'adaptation de cette méthode à notre problème et nous donnerons les résultats de cet algorithme.

Description de la Méthode Tabou

Cette méthode a été proposée par Glover (voir [27]). L'algorithme consiste à choisir à chaque transition, la meilleur solution voisine de la solution courante. Tant que l'on ne se trouve pas dans un optimum local, l'algorithme se comporte comme une procédure d'optimisation locale et améliore donc à chaque étape la valeur de la fonction objective. Lorsque l'on atteint un optimum local, on choisit le moins mauvais des voisins (c'est-à-dire celui qui donne l'accroissement le plus faible possible).

L'inconvénient que présenterait le choix d'un voisin quelconque serait que si un minimum local se trouve au fond d'une vallée profonde, il serait impossible de ressortir de celle-ci en

une seule itération, et un déplacement vers une solution de coût supérieur à la solution courante pourrait provoquer, à l'itération suivante, le déplacement inverse; on risquerait donc de cycler autour de ce minimum local. Pour éviter ce genre de phénomènes, on garde en mémoire les dernières solutions visitées et on interdit un retour vers celles-ci pour un nombre fixé d'itérations. En d'autre terme on conserve à chaque étape une liste de solutions "tabous" vers lesquelles il est interdit de se déplacer momentanément. Le but d'introduire des solutions tabous est de donner assez de temps à l'algorithme pour lui permettre de sortir de toute vallée contenant un minimum local.

On se donne en général un nombre maximum d'itérations $nbmax$ entre deux améliorations de la meilleure solution rencontrée comme condition d'arrêt. Dans le cas où on connaît une borne inférieure de la fonction objectif on arrête la recherche lorsqu'on atteint une solution proche de la borne inférieure.

Algorithme Tabou

1. Initialisation
   - T= solution quelconque ;
   - $nbiter = 0$; (itération courante)
   - $miter = 0$; (itération ayant donné la meilleure solution)
   - $T^* = T$; (meilleure solution)
   - $Lt = vide$; (Liste tabou)
   - initialiser la fonction d'aspiration A ;
2. Tant que ($B_* < \mathbb{E}(L_T)$) et (nbiter - miter < nbmax) faire

   (B_* est une estimation de l'optimum)
   - nbiter = nbiter+1 ;
   - Générer un ensemble V^* de solutions voisines de T ;
   - Choisir la meilleure solution T' de V^* tel que T' n'est pas dans Lt ou $\mathbb{E}(L_{T'} \leq A(\mathbb{E}(L_T)$
   - Mettre à jour la fonction aspiration A et la liste Lt ;
   - Si $\mathbb{E}(L_{T'}) \leq \mathbb{E}(L_{T^*})$ alors ($T^* = T'$; $miter = nbiter$; $T = T'$) ;

Adaptation de Tabou aux PVCPs

Il serait peu réaliste de considérer une liste de solutions tabous En effet, elle demanderait trop de place en mémoire et nous perdrions trop de temps à tester si une solution fait partie de Lt. Nous avons donc décidé de ne mémoriser qu'une information moins complète. Lorsqu'on fait un deux-inter-change des sommets i et j, on garde en mémoire seulement i et j et alors le couple (j, i) devient tabou, ce qui signifie que le deux-inter-change j et i n'est pas autorisé pour un nombre $|Lt|$ d'itérations. La taille $|Lt| = 10$ s'est expérimentalement avérée satisfaisante.

Il est parfois souhaitable de pouvoir retourner vers une solution tabou et ce afin d'explorer une nouvelle région de l'espace des solutions (puisqu'on ne retient qu'une information partielle de la solution, on peut donc interdire des solutions non visitées). Pour cette raison, on

V	p	*Espérance*	*Temps C.P.U*
25	0.7	3.820077	13.22
	0.5	3.366812	15.85
	0.1	1.161624	17.35
50	0.7	5.495785	270.10
	0.5	4.709543	285.07
	0.1	2.104695	374.92

Tableau 2.12 : Résultat de l'algorithme Tabou

		$TS + Structurecycle$		$TS + Shift$		$TS + deuxchange$	
V	p	Espérance	Temps C.P.U	Espérance	Temps C.P.U	Espérance	Temps C.P.U
23	0.7	3.7679	9.12	4.3291	12.86	3.7679	9.10
	0.1	1.0421	13.95	1.0499	12.27	1.0421	13.93
47	0.7	5.23	84.45	5.16	430.03	5.2452	275.05
	0.1	1.99	144.18	2.068	315.11	1.99	459.30

Tableau 2.13 : Comparaison des topologies pour TS

introduit un nouvel ingrédient appelé fonction d'aspiration et définie sur toutes les valeurs de la fonction objectif : lorsqu'une solution voisine de la solution courante fait partie de la liste de solutions tabous et satisfait la condition d'aspiration (c'est-à-dire $\mathbb{E}(L_{T'}) \leq A(\mathbb{E}(L_T))$, A est la fonction aspiration), on lève le statut tabou de cette solution et elle devient candidate lors de la sélection.

Nous avons défini la fonction aspiration comme étant égale à la valeur $\mathbb{E}(L_{T^*})$ de la meilleure solution.

Résultats numériques

On remarque que les algorithmes tabous sont nettement plus performant que ceux de type recuit simulé. Tabou, donne une qualité de solution comparable à celle du recuit simulé mais il requiert moins de temps de calcul.

2.4.4 Approximation de la Fonction Coût

D'après la Proposition 3, quand p est proche de 1, le tour optimal au sens PVC est une bonne heuristique pour notre problème et ceci revient à minimiser le premier terme de la fonctionnelle associée au PVCP. En remarquant que le $k^{\text{ème}}$ terme $(1-p)^k L_T^{(k)}$ de la fonctionnelle devient assez petit pour k assez grand ($p \leq 1$), on se propose d'approximer la fonctionnelle par les k premiers termes. Cette méthode nous permet d'avoir des interactions fixes entre les solutions voisines (indépendantes de n) et permettrait donc d'améliorer la complexité.

V	p	*Espérance final* v_1	*Temps C.P.U*
25	0.7	3.8054705	41.87
	0.5	3.2919015	37.15
	0.1	1.1532327	39.66
50	0.7	5.0254676	230.44
	0.5	4.4591916	230.77
	0.1	2.1652255	231.03

Tableau 2.14 : Temps d'exécution en second

V	p	v_0	v_1	v_{n-2}
25	0.7	3.830402	3.805470	3.804305
	0.1	1.154219	1.153232	1.152714
50	0.7	5.090686	5.025467	5.025467
	0.1	2.182250	2.165225	2.097167

Tableau 2.15 : comparaison des approximations

Pour $k \in [0..n-2]$ on définie v_k le tour qui résout le problème de minimisation suivant :

$$\mathbb{E}(L_{v_k}) = p^2 \sum_{r=0}^{k} (1-p)^r L_{T^r} \tag{2.41}$$

On a alors $v_0 = PVC$ et $v_{n-2} = PVCP$. Les problèmes intermédiaires n'ont pas une signification physique, mais on peut les utiliser pour trouver une bonne heuristique pour le PVCP. On trouve des bornes supérieures (voir Jaillet [34]) U_k pour chaque problème (2.42). Il parait naturel que la borne U_k soit meilleur que U_{k-1} ceci ne veut pas dire que $\mathbb{E}(L_{v_1}) \leq \mathbb{E}(L_{v_0})$ mais seulement en pire cas v_1 est plus proche de v_{n-2} que ne l'estv_0.

$$\mathbb{E}(L_{v_k}) - \mathbb{E}(L_{T_{PVCP}})/\mathbb{E}(L_{T_{PVCP}}) \leq U_k \tag{2.42}$$

Si on compare les Tableaux 2.14 et 2.7 on remarque qu'on trouve presque la même qualité de solution avec une diminution de 1/3 en temps d'exécution. On remarque d'après le Tableau 2.15 que la solution v_0 ne dépasse pas de 1% la solution v_{n-2} quand p proche de 1 ce qui est prévisible d'après la Proposition 3, mais même pour p petit la solution ne s'améliore que de 5% ce qui est très étonnant. Donc soit le recuit donne une mauvaise solution pour le PVCP ou la solution du PVC est une bonne approche pour le PVCP dans le cas où la répartition des points est uniforme.

On note que la valeur donnée pour v_{n-2} est le plus petit tour trouvé tout au long de l'algorithme par contre le tour final (obtenue qu'on on a un gel) a le même coût que v_1.

2.4.5 Algorithme de Partitionnement de Karp

On sait qu'il est très efficace de choisir des transformations pour lesquelles le voisinage de tout circuit ne contienne que des circuits de coûts peu différents. Comme notre fonction

V	p	*Espérance*	*Temps C.P.U*
25	0.7	5.873459	2.32
	0.5	3.994623	3.04
	0.1	0.371927	2.62
50	0.7	9.2825464	15.17
	0.5	6.9060659	15.02
	0.1	1.0344274	16.10

Tableau 2.16 : Résultat de l'algorithme de Karp pour les problèmes de petite taille

augmente avec la distance cette tâche n'est pas difficile. L'algorithme de *partitionnement* de Karp [46] nous permet d'avoir ce genre de propriété.

Description de l'algorithme de partitionnement de Karp

1- subdiviser le carré en m sous-carrés
chaque sous-carré contient au maximum t sommets
2- Construire le tour optimal dans
chaque sous-carrés.
3-choisir un représentant dans
chaque sous-carré.
4-construire le tour optimal à travers
ces représentants

Algorithme de partitionnement de Karp

L'algorithme de partitionnement de Karp consiste à subdiviser le carré en 2^k sous-carrés, chaque sous-carré contenant au maximum t sommets, puis on résout le problème dans chaque sous-domaine et on obtient ainsi les tours optimaux dans chaque domaine. On choisit ensuite un représentant dans chaque domaine et on construit le tour optimal à travers ces représentants. On obtient par cette procédure un chemin recouvrant (c'est un graphe connexe où tous les sommets sont de degré pair) enfin d'après [46] à partir duquel on peut déduire un tour à travers tous les points dont le coût est minimal.

On montre [46] que cet algorithme converge *asymptotiquement* vers la solution optimale du PVC euclidien, si les points $X^{(n)}$ sont indépendants et uniformément distribués dans $[0,1]^2$. Ce résultat n'est pas démontré pour le PVCP.

Résultats de KarpRS

On remarque d'après les tableaux 2.7 et 2.4.5 que l'algorithme de Karp donne une grande solution, pour $p > 0.5$ mais en un temps très court (de l'ordre de t plutôt que de l'ordre de n) par contre pour p petit le KarpRS donne une solution nettement meilleur que RS (deux fois plus petite). Cet algorithme devrait donner une assez bonne solution pour p proche de 1 car le PVC est une bonne approximation du PVCP.

V	p	$KarpRS$	$KarpTS$
2000	0.7	99.723339	99.023567
	0.5	73.354781	73.384876
	0.1	16.036738	15.136743
1000	0.7	48.300254	49.300890
	0.5	38.574696	38.571754
	0.1	8.0494698	8.0194567

Tableau 2.17 : Résultat de l'algorithme de Karp pour les problèmes de grande taille

2.4.6 Conclusion

L'algorithme de recuit simulé n'est pas particulièrement adapté au PVCP. L'algorithme tabou, facile à implémenter, donne une solution de qualité équivalente avec un temps d'exécution nettement plus réduit. Toutes les structures de voisinages, induisent une interaction totale entre deux solutions voisines et donc le calcul de la différence de coût se fait au meilleur des cas en $0(n)$. Les structures de voisinage, shift et deux-inter-change pour le PVCP(resp PVC) induisent un temps de calcul de la différence de coût en $0(n^2)$ (temps constant). Nous proposons une nouvelle topologie, structure de cycle, qui permet de ramener le calcul de la différence de coût à un ordre de $0(n)$. Nous remarquons que la solution du PVC est une bonne approche pour le PVCP euclidien pour p quelconque. Ceci étant prévisible pour p proche de 1 d'après la proposition 3, mais particulièrement surprenant pour p petit. Les résultats de la section 2.4.4 laisse penser que la solution du PVC euclidien est une très bonne approche pour le PVCP euclidien.

Malgré une implémentation optimale, tous ces algorithmes restent impraticables pour les problèmes de très grande taille. Dans les situations où on désire obtenir une solution à tous pris, on peut se contenter de l'algorithme de partitionnement de Karp combiné avec tabou. Cet algorithme converge très rapidement mais la solution généralement obtenue n'a pas une bonne qualité.

Chapitre 3

Problèmes d'Ordonnancement des Travaux Probabilistes

Les problèmes d'ordonnancement apparaissent dans un grand nombre de domaines : l'informatique (tâches : jobs ; ressources : processus ou mémoire). la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emplois du temps), etc.

On peut classer les problèmes d'ordonnancement en deux classes :

- Problèmes d'ordonnancement statique : si l'on connaît a priori toutes les données le concernant. Pour ce genre de problème l'étude est très vaste ([17],[18]).
- Problèmes d'ordonnancement dynamique : les informations (nombre de tâche, nombre de machines disponibles, date d'exécution des tâches...) sont soumises à des aléas ou sont prévisionnelles ou encore sont connues en probabilité. Les seules études probabilistes pour ces problèmes sont faites pour le cas où le temps d'exécution des tâches est une variable aléatoire exponentielle, plusieurs articles ([26] [14],...) ont développé cette version.

En supposant que la présence des tâches est connue seulement en probabilité, on obtient un problème d'ordonnancement dynamique (plus adapté à la réalité). En nous inspirant des travaux ([9],[35],[36],[37]) sur les problèmes d'optimisation combinatoire probabilistes (POCP) on proposera dans ce chapitre des stratégies pour ce problème.

Nous donnons au paragraphe suivant les définitions et les notations des problèmes d'ordonnancement classiques. Dans le paragraphe 2 nous donnons la définition des problèmes d'ordonnancement des travaux probabilistes (POTP). Dans le paragraphe 3 nous discutons les motivations et certaines applications du POTP. Nous commençons dans le paragraphe 4 par aborder les problèmes d'ordonnancement des travaux probabilistes à une machine, en particulier nous proposons une expression explicite de l'espérance d'un ordonnancement à une machine au sens POTP, puis nous étudions la complexité du problème. De la même façon nous traitons dans le paragraphe 5 le POTP à plusieurs machines, nous montrons que le problème est difficile et nous proposons des algorithmes exacts et certaines heuristiques, et nous étudions ensuite les propriétés combinatoires du POTP. Nous proposons au paragraphe 6 la stratégie de réaffectation pour le problème à plusieurs machines. Enfin nous donnons une généralisation à d'autres problèmes dans le paragraphe 7.

3.1 Définitions et Notations Générales des Problèmes d'Ordonnancement

Ordonnancer, c'est programmer dans le temps l'exécution d'un projet décomposable en tâches, en attribuant des ressources à ces tâches et en fixant en particulier leurs dates de début d'exécution tout en respectant des contraintes données. En nous inspirant de [12] nous allons donner une description générale des problèmes d'ordonnancement en introduisant les notations les plus couramment employées par les spécialistes.

Les différentes données d'un problème d'ordonnancement sont les tâches, les contraintes potentielles,les ressources et la fonction économique.

1. Tâches : elles sont le dénominateur commun des problèmes d'ordonnancement ; leurs définitions ne sont pas toujours évidentes (par exemple dans les problèmes du bâtiment, elles dépendent de la finesse du découpage : une tâche "menuiserie" peut être considérée comme un ensemble de tâches telles que : pose de la porte d'entrée, pose d'un placard, pose d'un parquet,...) et peuvent parfois sembler artificielles (par exemple dans les problèmes de transport, une tâche pourra très bien être l'occupation d'une portion de voie par un train).
   On suppose que le projet se décompose en n tâches. On notera t_i la date de début d'exécution de la tâche i et C_i sa date de fin d'exécution (ce sont des variables du problème à déterminer). On suppose que l'exécution d'une tâche i de durée p_i (en anglais :"processing time") peut commencer à la date r_i ("release date") et doit être achevée à la date d_i ("due date").

   On a donc pour les variables t_i et C_i les relations suivantes (ce sont des contraintes) :

   $$r_i \leq t_i \leq C_i \leq d_i,$$

   et si, de plus, la tâche ne peut être morcelée dans le temps (c'est à dire dans le cas non préemptif) nous avons :

   $$C_i = t_i + p_i.$$

   Enfin, dans certains cas, on a à considérer un "poids" w_i pour la tâche i qui pourra représenter une pénalité de retard si $C_i > d_i$ (par exemple, un coût de stock d'encours, etc).

2. Contraintes potentielles : les tâches peuvent être liées par des contraintes "potentielles" ; celles-ci englobent des contraintes de succession mais aussi des contraintes de localisation temporelle (telle tâche doit être achevée à telle date, ou, au contraire, son exécution ne doit pas commencer avant telle date).

3. Ressources : les tâches requièrent, pour être exécutées, certaines ressources qui peuvent introduire des contraintes de type "disjonctif" (deux tâches ne peuvent pas s'exécuter simultanément quand par exemple elles utilisent une même machine) ou des contraintes de type "cumulatif" (une généralisation des précédentes contraintes à plus de deux tâches à un moment donné). De plus, ces ressources peuvent être, soit renouvelables (machines, personnel), soit consommables (argent, matières premières). Lorsque les ressources sont des machines, elles peuvent être identiques ou différentes, dépendantes

ou non les unes des autres. Certaines ne sont disponibles qu'à certaines périodes, et certaines tâches peuvent requérir des machines spécifiques. Il peut y avoir d'autres ressources que les machines, par exemple de la main d'oeuvre, de l'argent, des fichiers informatiques. On connaît en général les quantités de ressources disponibles en fonction du temps.

4. Système préemptif : Dans certains cas, la réalisation d'une tâche peut être morcelée dans le temps, et on dit que le modèle étudié est préemptif (par exemple, dans un système multiprogrammé où l'on peut morceler l'exécution d'un job). Souvent la possibilité de "préempter" rend les problèmes plus faciles (parce que moins contraints) ; ainsi, pourra-t'on utiliser la préemption pour calculer les bornes de la fonction.

5. Fonction économique : le problème est de "programmer" ces tâches de façon à optimiser un certain objectif. La variables intervenant dans l'expression des fonctions économiques les plus utilisées est la date C_i de fin d'exécution de la tâche i.

   Les fonctions économiques les plus utilisées sont :

   - Durée totale ; la durée totale de l'ordonnancement, notée $Cmax$ est égale à la date d'achèvement de la tâche la plus tardive

   $$Cmax = \max_i C_i.$$

   - Minimisation d'un coût ; On s'intéressera particulièrement à ce genre de critère. Par exemple la moyenne pondérée des temps d'achèvement $\sum_i \omega_i C_i$ permet d'estimer le coût des stocks d'encours (en effet, la tâche i présente dans l'atelier entre les instant r_i et C_i, et, donc les stocks dont elle a besoin doivent être disponibles entre ces deux dates ; d'où un coût $\sum_i \omega_i(C_i - r_i)$ égale à une constante près à $\sum_i \omega_i C_i$). L'importance de ce critère est que sa minimisation réduit le nombre de tâches inachevées à chaque instant de l'ordonnancement ([18]).

   Pour la suite de l'étude nous introduisons les notations suivantes :

   a) Un ordonnancement σ des tâches $\{1, ..., n\}$ sur une machine est une permutation de $\{1, ..., n\}$ où $\sigma(i)$ est la $i^{\text{ème}}$ tâche affectée.

   b) Un ordonnancement ξ des n tâches sur m machines est une partition des tâches $\{1, ..., n\}$ en m classes $B_1, B_2, ...B_m$ (certaines toutefois peuvent être vides) où les tâches B_k sont celles exécutées par la machine k, et σ_i est l'ordonnancement des tâches B_i, on note alors $\xi = (\sigma_1, .., \sigma_i, ...\sigma_m)$.

   c) On notera M_i la machine sur laquelle i est exécutée, $j \in M_i$ c'est à dire j est exécutée sur la même machine que i. $|B_i|$ est le nombre de tâche affectées à la $i^{\text{ème}}$ machine

   d) Si les tâches sont indépendantes et non preemptive alors la fin d'exécution $C_{\sigma_k(i)}$ de la tâche $\sigma_k(i)$ est :

   $$C_{\sigma_k(i)} = \sum_{j \leq i} p_{\sigma_k(j)}$$

La moyenne des temps d'achèvement (mean finishing time) pour l'ordonnancement ξ :

$$mft(\xi) = \sum_{i=1}^{m} mft(\sigma_i)$$

avec

$$mft(\sigma_i) = \sum_{k=1}^{|B_i|} C_{\sigma_i(k)}$$

La moyenne pondérée des temps d'achèvement (weighted mean finishing time) est :

$$wmft(\xi) = \sum_{i=1}^{m} wmft(\sigma_i)$$

avec

$$wmft(\sigma_i) = \sum_{k=1}^{|B_i|} \omega_{\sigma_i(k)} C_{\sigma_i(k)}$$

3.2 Motivation et Application du POTP

Dans ce paragraphe on commencera par expliquer l'avantage de l'utilisation de la stratégie POTP, puis on donnera des exemples concrets où cette méthode est appliquée.

L'introduction de la stratégie a priori POTP dans des problèmes d'ordonnancement présente plusieurs motivations. La première correspond au désir de formuler et d'analyser des modèles qui sont plus appropriés pour des problèmes pratiques. En effet, il y a une quantité importante d'applications intéressantes pour les POTPs, notamment dans les contextes des systèmes de communications et de transport, des systèmes de prévision de gestion des stocks, des réseaux distribués d'ordinateurs, etc. La deuxième motivation est due au manque d'information au début : il est vrai qu'il est plus naturel de réoptimiser pour chaque exemplaire, mais ceci peut ne pas être possible faute de manque d'information au début ou du fait que le problème d'ordonnancement initial est $NP-complet$. La réoptimisation peut coûter très cher (et dans plusieurs domaines impossible), en effet les informations sont généralement très coûteuses.

On montre dans la Proposition 30 que le POTP peut être une bonne heuristique quand on n'a pas les moyens de réoptimiser. Une caractéristique importante du POTP est que la solution est plus robuste que celle du POT. Malheureusement le problème devient difficile. Une propriétés importante de la stratégie POTP est que l'ordre initial des tâches n'est pas perturbé par l'absence de certaines tâches. Cette propriété n'est pas vérifiée pour la stratégie de réoptimisation en effet l'absence d'une seule tâche perturbe totalement l'ordre initial (voir Paragraphe 3.6.3).

Dans beaucoup d'applications le POTP est la stratégie utilisée. le POTP s' applique à des domaines très divers on peut citer :

- Système multiprocesseur :

  De nombreux systèmes parallèles et distribués (par exemple, CM, Alliont, Warp, machines à transputer...) sont maintenant disponibles sur le marché. L'optimisation de l'ordonnancement et du placement des tâches dans ces systèmes, de l'allocation des

ressources et du routage des messages dans les réseaux de communication restent des problèmes mal compris et non résolus du point de vue pratique. Notre étude théorique de la stratégie POTP fournit un outils d'aide à la conception et à la réalisation des systèmes d'exploitation parallèles et distribués [50]. Par exemple un programme parallèle peut être vu comme une collection de tâches qui peuvent être exécutées de façon parallèle ou séquentielle. Ces tâches doivent être placées sur les processeurs d'une machine parallèle de manière à minimiser un certain objectif (temps maximal passé dans le système, moyenne des temps d'achèvement des tâches). Les deux programmes qui peuvent causer le "non-déterminisme" dans le parallélisme sont les branches conditionnelles et les boucles en effet la borne supérieure de la boucle peut être inconnue avant l'exécution du programme. Quand on est dans ce cas alors le système processeur parallèle doit se contenter d'une stratégie a priori et tenter d'ordonnancer les tâches pendant l'exécution. Un exemple plus concret : supposons qu'on veut "parallèliser" une boucle, les itérations de la boucle sont les tâches, on supposera qu'elles ont le même temps d'exécution donc on veut affecter les tâches aux différents processeurs. L'ordonnancement utilisé dans ce cas est : on divise les n itérations en $\lceil \frac{n}{m} \rceil$ étapes, dans chaque étape sauf peut être la dernière m itérations sont exécutées en parallèle. En supposant que la présence des itérations de la boucle ne sont pas indépendant alors la stratégie de réoptimisation est impossible par manque d'information donc on se contente d'une stratégie a priori.

- Guidage des véhicules :

  Trouver son chemin à toujours été pour le conducteur une préoccupation essentielle, depuis longtemps satisfaite par les cartes routières et la signalisation de direction. Trouver le meilleur itinéraire compte tenu de l'état du trafic (stratégie de réoptimisation) nécessite une connaissance des conducteurs des conditions de circulation et donc un équipement embarqué permettant au conducteur de recevoir des instructions aux points de choix, l'orientant vers sa destination. Cette méthode n'est pas réalisable (du moins pour le moment) et risque d'être très coûteuse, donc on utilise une stratégie a priori qui donne à l'usager une sorte de meilleur itinéraire moyen,(qui n'est pas toujours l'optimum). Pour modéliser ce problème on peut supposer que le véhicule i est présent avec une probabilité q_i alors la stratégie a priori POTP est une solution naturelle (avancer dans la file d'attente) du problème.

- Système de service :

  Considérons la salle des réservations de billets SNCF de la gare de Lyon à Paris un samedi de juin entre 10^h et 11^h du matin. Les préposés aux guichets ont des vitesses d'exécutions, courent des risques d'erreur et de fausse manoeuvre tous différents. Les deux ou trois cents clients qui vont être servis par des guichetiers durant la période d'observation ont des préoccupations diverses qui vont influencer leurs impatience plus ou moins aiguë.

  Supposons qu'on a n clients potentiels et m serveurs. Ces m guichets se comportent comme autant de sources de demandes pour l'ordinateur, comme celui ci ne peut traiter que N ($N \leq m$) réservations à la fois, on a entre les terminaux et l'ordinateur une nouvelle file d'attente. On a donc représenté le système de réservation de gare de Lyon

comme un réseau constitué par une juxtaposition de files d'attente.

La gestion de ces files d'attentes est généralement conditionnée par deux éléments : l'ordre dans lequel les clients attendent d'une part, la méthode de temporisation (si on permet aux clients de quitter la file sans être servis) d'autre part. On s'intéresse aux problème de temporisation. La temporisation consiste à éliminer les clients de la file après qu'ils aient attendus un temps donné. L'analyse de ce phénomène consiste à considérer qu'un client se décourage avec une certaine probabilité (la probabilité peut dépendre du temps d'attente). Alors en supposant que les terminaux sont dans des salles différentes notre stratégie POTP est une solution envisageable dans les files d'attentes des clients. De même on peut penser qu'un client annule sa réservation en cours avec une certaine probabilité d'erreur (due à des erreurs du clients), si le mécanisme de communication entre les processeurs est cher la stratégie POTP est donc une solution moins coûteuse que la stratégie de ré-optimisation.

3.3 Définition des Problèmes d'Ordonnancement de Travaux Probabilistes

On peut définir le problème générique suivant :

Le Problème d'Ordonnancement de Travaux Probabiliste.

Le problème d'ordonnancement de travaux (POT) que nous considérons est le suivant : m serveurs (machines) travaillant en parallèle doivent servir un ensemble V de n clients(tâches). Le temps de service requis par chaque client est connu d'avance : le client i prendra un temps p_i. Le problème est d'ordonnancer le service des clients afin de minimiser la moyenne du temps total passé dans le système, pour chaque client c'est à dire la quantité $mft = \sum_{i=1}^{n}$ (temps passé dans le système par le client i). Par la suite, seuls certains sous-ensembles $S_1, S_2, \ldots,$ des n clients devront réellement être servis, ces ensembles étant choisis successivement et indépendamment selon une probabilité définie sur 2^V. Le problème est de déterminer, avant de connaître la suite $(S_i)_i$, un ordre de service des n clients tel que, si S est le sous-ensemble à servir, les clients en question soient servis dans le même ordre que celui défini par l'ordonnancement a priori. La méthode de modification $\mathcal{U}$ consiste donc simplement à "gommer" de l'ordonnancement initial les clients qui n'appartiennent pas à S. Le problème de trouver un ordonnancement a priori minimisant l'espérance de la moyenne des temps totaux passés dans le système pour chaque client sous cette stratégie est appelé le problème d'ordonnancement de travaux probabiliste (POTP).

On étudiera en détail seulement les deux problèmes d'ordonnancement suivants :

- problèmes d'ordonnancement sans contraintes de ressources à une machine : La méthode de modification $\mathcal{U}$ consiste simplement à enlever de l'ordonnancement les tâches absentes.

- problèmes d'ordonnancement à plusieurs machines : On proposera deux méthodes de modification

  1. La stratégie POTP : $\mathcal{U}$ consiste à enlever les tâches absentes de chaque machine de l'ordonnancement initial à travers les n tâches.

2. La stratégie de réaffectation consiste à affecter les tâches présentes d'une façon optimale pour chaque exemplaire S. On note Σ_{Reafct} cette stratégie.

3.4 Problèmes d'Ordonnancement de Travaux Probabilistes à une Machine.

Espérance des Ordonnancements au sens POTP.

Théorème 11 *Soit n le nombre de tâches, on suppose que chaque tâche i a une probabilité de présence q_i, indépendante des autres tâches. L'espérance de l'ordonnancement σ est :*

$$I\!E(mft(\sigma)) = \sum_{i=1}^{n} q_{\sigma(i)} p_{\sigma(i)} (1 + \sum_{j>i} q_{\sigma(j)}). \tag{3.1}$$

Démonstration :

Sans perte de généralité supposons que $(1, 2, 3, ..., n)$ soit l'ordonnancement dont on veut évaluer la fonction objective. La durée p_i apparaît dans la fonction objective si et seulement si la tâche i est présente (probabilité q_i) ; dans ce cas elle apparaît en plus chaque fois qu'une tâche $j > i$ est présente (probabilité q_j). La contribution de p_i est alors $q_i(1 + \sum_{j=i+1}^{n} q_j)$. La fonction objective est donc $\sum_{i=1}^{n} q_{\sigma(i)} p_{\sigma(i)} (1 + \sum_{j>i} q_{\sigma(j)})$. ∎

Remarques :

- On note $mft(\sigma|n) = mft(\sigma|$ les n tâches sont présentes)
- Pour le cas où les tâches sont présentes avec la même probabilité q, si on note $K_n = \sum_{i=1}^{n} p_i$, la fonction objective de l'ordonnancement σ d'après (3.1) est :

$$I\!E(mft(\sigma)) = q^2 mft(\sigma|n) + q(1-q)K_n. \tag{3.2}$$

- On note N_i le nombre de tâches présente après i. $I\!E(N_{\sigma(i)}) + 1$ est égale à la contribution de $p_{\sigma(i)} q_{\sigma(i)}$: $I\!E(N_{\sigma(i)}) = \sum_{j>i} q_{\sigma(j)}$

Complexité du POTP à une Machine

Proposition 20 *Soit n le nombre de tâches, on suppose que la tâche i a une probabilité de présence q_i. Alors*

$$POTP \equiv POT. \tag{3.3}$$

Démonstration : On sait d'après [18] que la solution du POT est celle qui consiste à affecter

les tâches dans l'ordre croissant des dates d'exécutions. Supposons que $p_i \leq p_{i+1}\ \forall\, i \in [1,n]$. Nous avons donc $POT \equiv (1,2,3,...n)$.
Soit σ un ordonnancement quelconque on va montrer que

$$\mathbb{E}(mft(POT)) \leq \mathbb{E}(mft(\sigma)) \qquad \forall \sigma. \tag{3.4}$$

Soit $\sigma \neq POT$, alors

$$\exists\, j \qquad p_{\sigma(j+1)} \leq p_{\sigma(j)}, \tag{3.5}$$

on considère l'ordonnancement σ^j construit à partir de σ en échangeant les $j^{\text{ème}}$ et $(j+1)^{\text{ème}}$ tâches alors

$$\mathbb{E}(mft(\sigma^j)) - \mathbb{E}(mft(\sigma)) = q_{\sigma(j)}q_{\sigma(j+1)}(p_{\sigma(j+1)} - p_{\sigma(j)}) \tag{3.6}$$

D'après (3.5) et (3.6) on a

$$\mathbb{E}(mft(\sigma^j)) - \mathbb{E}(mft(\sigma)) \leq 0. \tag{3.7}$$

Par un nombre fini de réarrangements on obtient le résultat. ∎

Remarque :D'après la Proposition 1 les POTs sont stables : Soient σ l'ordonnancement optimal à travers les n tâches et $S \subset \{1,...,n\}$, alors la solution optimale à travers S est obtenue simplement en enlevant de σ les tâches qui ne sont pas dans S. Ceci est dû au fait que la solution optimale du POT est obtenue par la méthode gloutonne (ordonner les tâches par ordre croissant de service), comme le rangement n'est pas perturbé par la modification, cette solution reste valable pour le POTP. Donc la " stratégie a priori " est équivalente à la "stratégie de réoptimisation".

3.5 Problèmes d'Ordonnancement à plusieurs Machines

Nous supposons ici que :

- Les tâches sont indépendantes (il n'y a pas de relation d'antériorité).
- On a m machines, chaque machine peut effectuer chaque tâche,
- chaque tâche i est présente avec une probabilité q_i.
- P est la matrice des durées des n tâches sur les machines (notation p_i^k = durée de la tâche i sur la machine k. Si les machines sont identiques alors $p_i^k = p_i$).

Définition du POTP à plusieurs Machines.

Un ordonnancement ξ des n tâches sur m machines est une partition des tâches $\{1,...,n\}$ en m classes $B_1, B_2, ...B_m$ où les tâches B_k sont celles exécutées par la machine k, et σ_i est l'ordonnancement des tâches B_i. La méthode de modification $\mathcal{U}$ consiste alors simplement à "gommer" dans chaque ordonnancement σ_i les tâches absentes de B_i.

Espérance des Ordonnancement à plusieurs Machines au sens POTP

Corollaire 6 *Soient n le nombre de tâche, m le nombre de machines. on suppose que toutes les tâches ont la même probabilité de présence q. L'espérance de l'ordonnancement ξ est :*

$$\mathbb{E}(mft(\xi)) = q^2 mft(\xi|n) + q(1-q)K_n. \tag{3.8}$$

Corollaire 7 *Soient n le nombre de tâches, m le nombre de machines, on suppose que la tâche i est présente avec une probabilité q_i. L'espérance de la fonctionnelle pour l'ordonnancement $\sigma = (\sigma_1, ..., \sigma_m)$ est :*

$$\mathbb{E}(mft(\xi)) = \sum_{i=1}^{m} \mathbb{E}(mft(\sigma_i)). \tag{3.9}$$

Remarques :

- Ces corollaires se déduisent directement de la définition de la fonctionnelle qui est additive.
- Le résultat 3.9 est général : Pour le cas où les machines sont identiques d'après (3.1)

$$\mathbb{E}(mft(\sigma_k)) = \sum_{i=1}^{n} q_{\sigma_k(i)} p_{\sigma_k(i)} (1 + \sum_{j>i} q_{\sigma_k(j)})$$

  Pour le cas où les machines ne sont pas identiques, il suffit de changer dans l'équation ci dessus $p_{\sigma_k(i)}$ *par* $p^k_{\sigma_k(i)}$.

$$\mathbb{E}(mft(\sigma_k)) = \sum_{i=1}^{n} q_{\sigma_k(i)} p^k_{\sigma_k(i)} (1 + \sum_{j>i} q_{\sigma_k(j)})$$

Complexité du POTP à plusieurs Machines

D'après le Corollaire 6 le POT à plusieurs machines résout le POTP lorsque les tâches ont la même probabilité de présence. Malheureusement ce résultat n'est pas vrai dans le cas d'une distribution quelconque. En effet considérons l'exemple suivant :
Exemple : $n = 3$, $m = 2$, $p_i = i \ \forall \ i$ et $q_3 = 1/6$, $q_2 = 1/4$, et $q_1 = 1$. Alors
POTP $= \{1\} \cup \{2,3\}$ et POT $= \{1,3\} \cup \{2\}$.
On sait que les POTs à plusieurs machines sont solubles en un temps polynomial d'après [11], donc d'après le Corollaire 6 il en est de même pour les POTPs a distribution de Bernoulli (toutes les tâches ont la même probabilité de présence). Une question se pose : dans le cas d'une distribution quelconque le POTP reste-t-il un problème facile ?

Théorème 12 *POTP est NP complet.*

Démonstration : Il est clair que le POTP appartient à la classe NP, en effet pour un ordonnancement donné ξ on peut calculer $\mathbb{E}(mft(\xi))$ en un temps polynomial et le comparer avec une borne B.
On considère P_1 le problème de minimisation de délai d'exécution pondéré suivant :

- Pas de relation d'antériorité entre les n tâches.
- La pondération $w_i = q_i \ \forall$ i
- La durée de l'exécution de la tâche i est $q_i p_i$, la date de début d'exécution de la tâche i est t_i :

$$t_{\sigma_k(i)} = \sum_{j<i} q_{\sigma_k(j)} p_{\sigma_k(j)}.$$

La fonction objective de P_1 pour un ordonnancement ξ est :

$$wmft_{P_1}(\xi|n) = \sum_{k=1}^{m} \sum_{i=1}^{|B_k|} w_{\sigma_k(i)} C_{\sigma_k(i)} = \sum_{k=1}^{m} \sum_{i=1}^{|B_k|} q_{\sigma_k(i)} t_{\sigma_k(i)} + \sum_{i=1}^{n} q_i^2 p_i.$$

comme $\sum_{i=1}^{n} q_i^2 p_i$ est une constante, le problème revient à minimiser

$$\sum_{k=1}^{m} \sum_{i=1}^{|B_k|} q_{\sigma_k(i)} t_{\sigma_k(i)} = \sum_{k=1}^{m} \sum_{i=1}^{|B_k|} q_{\sigma_k(i)} \sum_{j<i} q_{\sigma_k(j)} p_{\sigma_k(j)}.$$

Or l'espérance au sens POTP de l'ordonnancement ξ est égale à une constante près à la fonction objective de P_1. En effet d'après (3.9)

$$\begin{aligned} \mathbb{E}(mft(\xi)) - \sum_{i=1}^{n} q_i p_i &= \sum_{k=1}^{m} \sum_{j=1}^{|B_k|} q_{\sigma_k(j)} p_{\sigma_k(j)} \sum_{j<i} q_{\sigma_k(i)} \\ &= \sum_{k=1}^{m} \sum_{i=1}^{|B_k|} q_{\sigma_k(i)} \sum_{j<i} q_{\sigma_k(j)} p_{\sigma_k(j)}. \end{aligned}$$

Donc le POTP se ramène (en un temps polynomial) à P_1. P_1 étant un problème NP complet d'après [11], il en est de même du POTP. ∎

Remarque :L'ordre de priorité du POTP est celui du POT, en effet d'après la démonstration précédente le POTP se ramène à P_1, [11] l'ordre de priorité du problème P_1 est : $r_i = \dfrac{q_i}{q_i p_i} = \dfrac{1}{p_i}$

Relation entre le POTP et le POT

Proposition 21

$$\frac{\mathbb{E}(mft(POT)) - \mathbb{E}(mft(POTP))}{\mathbb{E}(mft(POTP))} \leq (\frac{q_{max}}{q_{min}} - 1)[1 + \frac{q_{max}}{q_{min}} - \frac{1}{n}] \qquad (3.10)$$

$$\frac{mft(POTP|n) - mft(POT|n)}{mft(POT|n)} \leq (\frac{q_{max}^2}{q_{min}^2} - 1) \qquad (3.11)$$

où $q_{max} = \max_i q_i$ *et* $q_{min} = \min_i q_i$

Démonstration : Soit ξ un ordonnancement on a d'après (3.9)

$$\mathbb{E}(mft(\xi)) = \sum_{k=1}^{m} \sum_{i=1}^{|B_k|} q_{\sigma_k(i)} p^k_{\sigma_k(i)} (1 + \sum_{j>i} q_{\sigma_k(j)}) \tag{3.12}$$

On majore les q_i par q_{max}

$$\mathbb{E}(mft(POT)) \leq q^2_{max} mft(POT|n) + q_{max}(1 - q_{max}) K_n \tag{3.13}$$

POT est l'ordonnancement optimal on a d'après (3.13)

$$\mathbb{E}(mft(POT)) \leq q^2_{max} mft(POTP|n) + q_{max}(1 - q_{max}) K_n \tag{3.14}$$

Alors comme $q_{min} \leq q_i$

$$q^2_{min} mft(\xi|n) + q_{min}(1 - q_{min}) K_n \leq \mathbb{E}(mft(\xi)) \tag{3.15}$$

On a d'après (3.14) et (3.15)

$$\mathbb{E}(mft(POT)) \leq \frac{q^2_{max}}{q^2_{min}} \mathbb{E}(mft(POTP)) + q_{max} K_n (1 - \frac{q_{max}}{q_{min}}). \tag{3.16}$$

Comme $1 - \frac{q_{max}}{q_{min}} \leq 0$ et $\frac{1}{n} \leq \frac{q_{max} K_n}{\mathbb{E}(mft(POTP))}$ on a le résultat.

$$\frac{\mathbb{E}(mft(POT)) - \mathbb{E}(mft(POTP))}{\mathbb{E}(mft(POTP))} \leq (\frac{q^2_{max}}{q^2_{min}} - 1) + \frac{1 - \frac{q_{max}}{q_{min}}}{n} \tag{3.17}$$

∎

Remarques :

- La borne est atteinte seulement dans le cas où $q_{max} = q_{min}$ sinon la majoration de l'équation (3.14) est stricte.
- On remarque d'après la proposition précédente que le POT est une assez bonne solution quand $q_i > 1/\sqrt{2}$. Mais quand $\min q_i \to 0$ on n'a plus d'information.

Proposition 22

$$\mathbb{E}(mft(POTP)) \leq \mathbb{E}(mft(POT)) \leq m \frac{n+1}{n+m} \mathbb{E}(mft(POTP))$$

Démonstration :

On a vu dans le théorème 12 que notre problème (POTP) se ramène a un problème de minimisation de délai d'exécution pondéré P_1. Soit C_1 le coût optimal quand $m = 1$ (si $p_1 \leq ... \leq p_n$ alors $C_1 = \sum_{i=1}^{n} q_i (\sum_{j=1}^{i} q_j p_j)$ on a alors (voir [20])

$$\frac{m+n}{m(n+1)} C_1 \leq wmft_{P_1}(POTP|n) \leq C_1$$

Or l'ordonnancement de coût C_1 est le pire des cas donc

$$wmft_{P_1}(POT|n) \leq C_1$$

Donc

$$\mathbb{E}(mft(POT)) \leq m\frac{n+1}{n+m}\mathbb{E}(mft(POTP))$$

∎

On remarque d'après les deux propositions précédentes que pour m constant ou q_{min} pas trop petit que la solution déterministe est une heuristique en pire cas constant $\frac{\mathbb{E}(mft(POT))}{\mathbb{E}(mft(POTP))} = O(1)$. On va donner un exemple où le POT peut être une faible solution pour le POTP quand m tend vers l'infini et $q_{min} = O(1/m)$.

Exemple

- Soit $n = m^2$
- si $i \ mod \ m \neq 0$, $j \ mod \ m \neq 0$ et si $\lfloor \frac{i}{m} \rfloor = \lfloor \frac{j}{m} \rfloor$ alors $p_i = p_j$ et $q_i = q_j = q_1$
- si $i \ mod \ m = 0$ alors $p_i = p_{i-1} + \epsilon$ et $q_i = 1$
- $p_i = i$ si $i \ mod \ m = 1$.

Il est clair que $POT \equiv (POT_1, ... POT_i, ...POT_m)$ où $POT_i(j) = (j-1)m + i$ pour $1 \leq j \leq m$.

On note $K_1 = \sum_{i=1}^{m} p_{(i-1)m+1}$ sachant que les tâches sont présentes avec une même probabilité q_1 sur les machines j pour $1 \leq j \leq m-1$ alors on a d'après (3.2) :

$$\mathbb{E}(mft(POT_j) = q_1^2 mft(POT_1|n) + q_1(1-q_1)K_1$$

Puisque sur la $m^{\text{ème}}$ machine toutes les tâches sont présentes alors

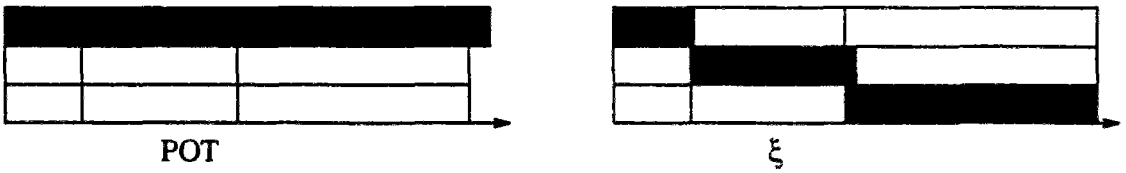
$$\mathbb{E}(mft(POT_m) = mft(POT_m|n) = mft(POT_1|n) + \epsilon\frac{m(m+1)}{2}$$

D'après les deux égalités ci dessus on a

$$\begin{aligned}\mathbb{E}(mft(POT)) &= mft(POT_1|n) + (m-1)q_1^2 mft(POT_1|n) \\ &+ (m-1)q_1(1-q_1)K_1 + \epsilon\frac{m(m+1)}{2}\end{aligned}$$

On considère la solution ξ qui consiste a répartir les m tâches toujours présentes sur les m machines : on échange dans la solution POT la $i^{\text{ème}}$ tâche exécutée sur la $m^{\text{ème}}$ machine par la $i^{\text{ème}}$ tâche affectée à la $i' = m - i + 1$ machine.

Pour calculer le coût de ξ on change la tâche toujours présente im dans ξ par une tâche présente avec une probabilité q_1 et de temps d'exécution p_{im-1} on obtient un problème où



toutes les tâches ont même probabilité, soit ξ' l'ordonnancement de ce nouveau problème donc d'après (3.2) on a pour $1 \leq k \leq m$:

$$\mathbb{E}(mft(\xi_k^{'})) = q_1^2 mft(POT_1|n) + q_1(1-q_1)K_1 \tag{3.18}$$

On peut caclculer la différence de coût des ordonnancements $\xi_{i'}$ et $\xi_{i'}^{'}$:

$$\begin{aligned}\mathbb{E}(mft(\xi_{i'})) - \mathbb{E}(mft(\xi_{i'}^{'})) &= q_1(m-i)p_{im} - q_1^2(m-i)p_{im-1} \\ &+ (p_{im} - q_1 p_{im-1}) \\ &+ q_1(1-q_1)\sum_{j=0}^{i-1} p_{jm+i'}\end{aligned}$$

En sommant sur toutes les machines on obtient :

$$\begin{aligned}\mathbb{E}(mft(\xi)) &= m[q_1^2 mft(POT_1|n) + q_1(1-q_1)K_1] \\ &+ q_1(1-q_1)mft(POT_1|n) + q_1\epsilon\frac{m(m+1)}{2} \\ &+ (1-q_1)K_1 + m\epsilon \\ &+ q_1(1-q_1)(mft(POT_1|n) - K_1)\end{aligned}$$

Donc

$$\begin{aligned}\mathbb{E}(mft(\xi)) &= 2q_1 mft(POT_1|n) + (m-2)q_1^2 mft(POT_1|n) \\ &+ (1-q_1)K_1 + m\epsilon \\ &+ q_1(1-q_1)(m-1)K_1 + q_1\epsilon\frac{m(m+1)}{2}.\end{aligned}$$

Donc si $q_1 = O(1/n)$ et m tend vers l'infini alors

$$\mathbb{E}(mft(POT))/\mathbb{E}(mft(POTP)) = O(m).$$

Cas particuliers

On va donner des cas particuliers où le POTP se résout polynomialement.

Proposition 23

$$E[\Sigma_{Reafct}] < I\!E(mft(POTP)) \ \ si\ m < n \tag{3.19}$$

$$E[\Sigma_{Reafct}] \equiv I\!E(mft(POTP)) \ \ si\ m = n\ ou\ m = 1 \tag{3.20}$$

Démonstration :

- m=1 où $m = n$, on a la propriété suivante :

  $(\mathcal{P}1)$ si ξ^* est la solution optimale du problème, soit S un sous ensemble des tâches alors la solution optimale à travers S est obtenue en enlevant les tâches qui ne sont pas dans S de ξ^*.

- m < n. La propriété $(\mathcal{P}1)$ n'est pas vérifiée pour m> 1, en effet il suffit de prendre $S = (i, j)$ où i et j sont effectuées sur la même machine (un tel couple existe car m $< n$).

∎

Proposition 24 *Supposons que seule la tâche 1 est présente avec une probabilité q_1, les autres tâches sont toujours présentes, alors le problème se résout polynomialement.*

Démonstration : Nous allons montrer qu'on peut construire un graphe $G = (X, Q)$ qui a la propriété d'admettre un flot maximal de coût minimal si et seulement si il existe un ordonnancement de coût minimal. Si la tâche i est suivie de s tâches sur la machine k sa durée contribue à la valeur de l'espérance, d'après (3.1), par la quantité $(s' + 1)q_i p_i^k$ avec $s' = s$ si 1 n'est pas parmi les k successeurs sinon $s' = s - 1 + q_1$. On considère alors la matrice Q à n colonnes et $(2n - 1)m$ lignes définie par

$$Q_{li} = (s' + 1)q_i p_i^k \quad si \quad l = sm + k \quad 1 \leq s \leq n \quad 1 \leq k \leq m \tag{3.21}$$

La matrice des coûts Q de ce problème est donnée par la Figure 1, avec P' la matrice $(q_i p_i^k)_{i \in T, k \in [1,m]}$.

$Q =$

m	P'
2m	$2P'$
	$(q_1 + 1)P'$
	nP'
(2n-1)m	$(q_1 + n - 1)P'$

Fig 1

Le réseau est représenté sur la Figure 2. l'arc (l, i) de ce réseau est alors valué par Q_{li}, la capacité des arcs est égale à 1 sauf celle de l'arc de retour égale à n.

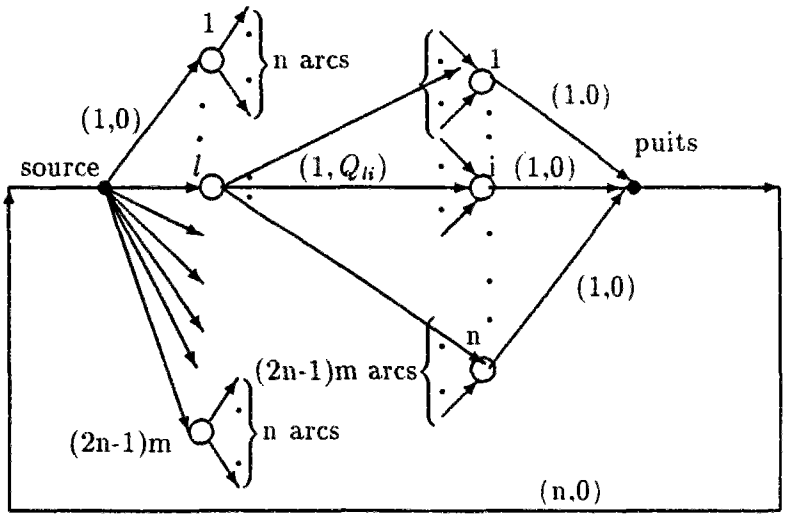


Figure 2

A un ordonnancement réalisable ξ, il est aisé de faire correspondre un flot maximal f_ξ ; en effet d'après le théorème de la moyenne [54], comme la capacité des arcs est entière alors le flot est à valeur entière, donc il prend les valeurs 0 ou 1 ; or si la tâche i est exécutée sur la machine k dans l'ordonnancement ξ et admet s_ξ successeurs sur cette machine, on affecte l'arc (l_ξ, i) où $l_\xi = s_\xi m + k$ ainsi que les arcs $(source, l_\xi) et (i, puits)$ d'un flux unité, sinon d'un flux nul.ξ et f_ξ ont le même coût.

On remarque qu'il existe des flots maximaux où on n'a pas d'ordonnancement correspondant. Si on considère le flot maximal de coût minimal défini sur ce réseau, il existe un ordonnancement réalisable correspondant ; en effet si le flux de l'arc $(l, i) = 1$ $(l = s'm + k)$ les sommets l_1 tel que $l_1 < l$ (c'est à dire $s'_1 < s'$) sont saturés (il existe j tel que le flux de l'arc (l_1, j) est saturé), sinon :

- on annule le flux des arcs $(source, l), (l, i) et (i, puits)$
- on fait passer à 1 le flux des arcs $(source, l'), (l', i) et (i, puits)$.

on aura un nouveau flot de coût inférieur, en effet la variation de coût est alors $(s' + 1)p_i^k - (s'_1 + 1)p_i^k = (s' - s'_1)p_i^k > 0$, ce qui contredit l'hypothèse d'optimalité du flot initial.

La résolution du problème de flot maximal de coût minimal de matrice Q, se résout polynomialement (en fonction de la dimension de la matrice Q) par l'algorithme de *Ford* et *Fulkerson* (1956) [22]. ∎

Remarques : La Proposition 24 se généralise pour les deux cas suivant :

1. Si la tâche 1 a une probabilité q_1 et les autres tâches ont la même probabilité de présence q $q \neq q_1$, le problème reste polynomial, en effet il suffit de remplacer dans la démonstration précédente si la tâche i est suivie de s tâches sur la machine k sa durée contribue à la valeur de l'espérance, d'après (3.1), par la quantité $(s' + 1)q_i p_i^k$ avec $s' = sq$ si 1 n'est pas parmi les k successeurs sinon $s' = sq - 1 + q_1$. On montre de la même façon que le problème se ramène à un problème de flot maximal de coût minimal. Donc si on a un nombre fixe k (indépendant de n) de probabilité différentes le problème est polynomial.

2. Si k est le nombre de tâches présentes avec une certaine probabilité, on peut montrer de la même façon que le problème se ramène à un problème de flot maximal de coût minimal de matrice Q. Mais la matrice Q serait alors de dimension $2^k.m$ et donc pour k= $O(n)$ l'algorithme de *Ford et Fulkerson*(1956) n'est plus polynomial.

Proposition 25 *Supposons que seul un sous ensemble de k (indépendant de n) tâches est présent avec une probabilité, les autres tâches sont toujours présentes, alors le problème se résout polynomialement.*

Proposition 26 *Si les $q_i p_i$ sont tous égaux pour $1 \leq i \leq n$, alors le POTP se résout polynomialement.*

Démonstration : On a vu que notre problème se ramène à $P1$. Or la solution optimale de $P1$ si les tâches ont le même temps d'exécution est donnée par l'algorithme suivant :

1. Ordonner les tâches dans l'ordre des pondérations (ce qui correspond dans notre cas aux probabilités q_i) les plus grandes.
2. Affecter les m tâches les plus prioritaires non affectées aux m différentes machines jusqu'a ce que toutes les tâches soient affectées.

La complexité du problème est $O(nlogn)$. ∎

Algorithmes Exacts et Approximatifs pour le POTP à Machines identiques.

On a montré dans le paragraphe ci dessus que le POTP est NP-complet, on va présenter un algorithme exact de type programmation dynamique qui donne la solution optimale, on proposera ensuite plusieurs heuristiques. La première heuristique est un schéma d'approximation basé sur des modifications dans l'algorithme exact, on étudiera l'efficacité de cette approche. On proposera ensuite plusieurs algorithmes de liste : dans ces méthodes une heuristique permet de classer les tâches selon un ordre de priorité puis on affecte les tâches de plus haute priorité. Ces méthodes sont performantes expérimentalement, malheureusement l'étude mathématique de leurs efficacité paraît difficile.

Algorithme de Programmation Dynamique pour le POTP On a montré que le POTP dans le cas de probabilités différentes et les machines sont identiques se ramène à un problème de minimisation de délai d'exécution pondéré P_1. Sachant que dans le cas d'une seule machine la stratégie optimale consiste à affecter les tâches par ordre croissant de leur rapport de priorité $r_i = 1/p_i$ voir [18], cet ordre est noté $WSPT$ (*weighted shortest processing time*), un ordonnancement peut être vu comme une partition de l'ensemble des travaux en m sous ensembles.
On va présenter une approche du type programmation dynamique proposée dans [58].
Soit I l'ensemble des n tâches et J un sous ensemble de I. Soit $J' = I - J$ le complémentaire de J. D'après le principe d'optimalité, les tâches dans J doivent être ordonnancées de façon

optimale indépendamment de l'ordre des tâches dans J'. On note $POTP_J$ l'ordonnancement optimal à travers J, on a donc la formule suivante :

$$wmft_{P_1}(POTP_J) = \min_{k \in J}(wmft_{P_1}(POTP_{J\setminus\{k\}}) + contribution(k)) \qquad (3.22)$$

On se restreint au cas $m = 2$. On donnera après la généralisation à plus de deux machines.

Définition 2 *[58] Un ordonnancement partiel $\xi^i = (U, V)$ est une partition des i premières tâches en deux ensembles U et V. Le complémentaire de ξ^i est une partition des tâches non affectées en deux sous ensembles X,Y tel que la partition $(U \cup X,\ V \cup Y)$ définit un ordonnancement des n travaux.*

On note $S^{(i)}$ l'ensemble de tous les ordonnancements partiels ξ^i tel que leurs complémentaires peuvent donner un ordonnancement optimal. Les éléments de $S^{(i)}$ seront des couples (T,t) où T est le coût de l'ordonnancement partiel correspondant à la partition et t est le temps d'achèvement de l'ordonnancement partiel sur la machine 1 (le temps d'achèvement sur la machine 2 est $\sum_{j=1}^{i} q_j p_j - t$). Le couple (T,t) $\in S^{(i)}$ représente la contribution des travaux 1,2,...i dans l'ordonnancement total. En effet : Soit (T, t) un couple $\in S^{(i)}$ correspondant à l'ordonnancement partiel $\xi^i = (U, V)$. Soient X et Y une partition du reste des travaux, $X = (\bar{x}_1, ..., \bar{x}_l)$, $Y = (\bar{\bar{y}}_1, ..., \bar{\bar{y}}_h)$. Soient $w_1 = \sum_{j \in X} \bar{q}_j$ et $w_2 = \sum_{j \in Y} \bar{\bar{q}}_j$. Soit $T^{(i)} = \sum_{j=1}^{i} q_j p_j$ (la somme des temps d'exécution des tâches affectées). Alors le coût correspondant à l'ordonnancement $(U \cup X, V \cup Y)$ est :

$$wmft_{P_1}(U \cup X, Y \cup V) = T + T'_x + T'_y + w_1 t + w_2(T^{(i)} - t).$$

où $T'_x = \bar{q}_1(\bar{q}_1\bar{p}_1) + ... + \bar{q}_l(\bar{q}_1\bar{p}_1 + ... + \bar{q}_l\bar{p}_l)$ et $T'_y = \bar{\bar{q}}_1(\bar{\bar{q}}_1\bar{\bar{p}}_1) + ... + \bar{\bar{q}}_l(\bar{\bar{q}}_1\bar{\bar{p}}_1 + ... + \bar{\bar{q}}_h\bar{\bar{p}}_h)$.

On va présenter des règles qui vont nous permettre de maintenir seulement les ordonnancements partiels de coût distincts et de temps d'achèvement distincts.

Lemme 3 *[58] Soient deux couples (T_1, t_1) et (T_2, t_2), $t_1 = t_2$ ou $t_1 = T^{(i)} - t_2$, si $T_1 < T_2$ alors (T_2, t_2) peut être éliminer de $S^{(i)}$;*

Lemme 4 *[58] Soient deux couples (T, t_1) et (T, t_2), $t_1 \leq t_2$ alors :*

- *si $t_1, t_2 \leq T^i/2$ alors (T, t_1) peut être éliminé de $S^{(i)}$;*
- *si $t_1, t_2 \geq T^i/2$ alors (T, t_2) peut être éliminé de $S^{(i)}$;*
- *si $t_1 < T^i/2$ et $t_2 > T^i/2$ alors :*

  si $(t_1 + t_2) > T^i$ (T, t_1) peut être éliminé de $S^{(i)}$
  sinon (T, t_2) est éliminé de $S^{(i)}$

Preuve voir [58].

L' algorithme de programmation dynamique pour le POTP à probabilités différentes (APDPOTP) est construit en deux étapes : Générer les n ensembles $S^{(i)}$ où les $S^{(i)}$ sont obtenus à partir de $S^{(i-1)}$, ensuite chercher la solution optimale.
Pour simplifier l'élagage on utilise la symétrie du problème, ainsi on considère les couples $(T,t) \in S^{(i)}$ où t est le temps passé sur la machine la plus engagée ($t \geq T^i/2$). On introduit une liste chaînée d'éléments binaires e pour indiquer si le couple courant représente un changement de machine du couple duquel il est obtenue dans $S^{(i-1)}$. Bien que la chaîne e n'est pas nécessaire pour calculer le coût optimal, elle est utile pour obtenir la solution optimale (pour remonter dans les ensembles $S^{(i)}$). Les éléments dans $S^{(i)}$ sont donc des triplets (T,t,e) avec le $i^{\text{ème}}$ terme de e nul si et seulement si il n'y a pas de changement de machine.

Algorithme APDPOTP

- données :$(q_i,p_i)1 \leq i \leq n$ avec $p_1 \leq ... \leq p_n$. résultat : un uplet d'ordonnancement optimal.
- initialiser $S^{(1)} = \{(q_1^2p_1, q_1p_1, 0)\}$, T' est la somme des temps d'exécution des tâches affectées $T' = q_1p_1$
- étape 1 { générer les $S^{(i)}$ }
  Pour i=2..n
  1. $A = \emptyset, C = \emptyset, D = \emptyset$
  2. **Pour** chaque triplet $(T,t,e) \in S^{(i)}$
     - (exécuter i sur la machine la plus engagée)
       $A = A \cup \{(T + q_i(t + q_ip_i), t + q_ip_i, e + 0)\}$
     - (exécuter i sur la machine la moins engagée)
       si $T' - t + q_ip_i > t$ alors $C = (T + q_i(T' - t + q_ip_i), T' - t + q_ip_i, e + 1) \cup C$
       sinon $D = D \cup \{(T + q_i(T' - t + q_ip_i), t, e + 1)\}$ (fin de si)
  3. (mettre à jour T') $T' = T' + q_ip_i$
  4. réunir C et D dans un ordre croissant de la deuxième coordonnée pour avoir B.
  5. réunir A et B pour obtenir $S^{(i)}$. Puisque A et B sont ordonnés suivant la deuxième coordonnée alors il en est de même pour $S^{(i)}$.
  6. (élagage de $S^{(i)}$) **Pour** tous les triplets $(T_1,t,e_1),(T_2,t,e_2) \in S^{(i)}$ *et* $T_1 \leq T_2$ alors $S^{(i)} = S^{(i)} - \{(T_2,t,e_2)\}$
- étape 2 { Obtenir la solutions optimale }
  La solution optimale correspond à l'élément (T,t,e) de $S^{(i)}$ ayant le plus petit T. L'ordonnancement est obtenu par décodage de la chaine binaire voir [32].

Complexité de l'Algorithme APDPOTP Le nombre de couples dans $S^{(i)}$ est borné par 2^i puisque le nombre de couples ne peut que doubler à chaque étape. D'autre part comme $t \leq M = \sum_{j=1}^{n} q_jp_j$, $S^{(i)}$ ne peut pas contenir plus que M *uplets* (d'après la règle d'élimination). Les réunions se font en un temps linéaire en fonction de la taille de $S^{(i)}$ (car les ensembles sont ordonnés). Ainsi le nombre total pour les n itérations est $O(min(2^n, nM))$ puisque $\sum_{i=1}^{n} 2^i = O(2^n)$.

Remarque :
Pour M petit, l'algorithme APDPOTP donne la solution optimale en un temps polynomial (par exemple $p_i = i/q_i$ la complexité de l'algorithme est $O(n^3)$). Pour M très grand le temps de calcul est exponentiel (même le stockage est exponentiel).

Généralisation à plusieurs machines La généralisation de APDPOTP à m machines se fait en considérant $m+1$ uplets voir[33],

- la première coordonnée de l'uplet représente le coût de l'ordonnancement partiel correspondant à la partition.
- La $k^{\text{ème}}$ coordonnée de l'uplet représente le temps t_k d'achèvement de l'ordonnancement partiel sur la machine k. (le temps d'achèvement sur la dernière machine est $\sum_{j=1}^{i} q_j p_j - \sum_{k=1}^{m-1} t_k$).

$S^{(i)}$ est génèré à partir de $S^{(i-1)}$ en ajoutant m *uplets*, la complexité de l'algorithme est $O(min(m^n, nM))$. Cet algorithme n'est pas aussi efficace que dans le cas $m = 2$.

Heuristiques pour le POTP

Schéma d'approximation La programmation dynamique peut conduire à des temps de calcul et à des encombrements en mémoire excessifs. Pour améliorer son efficacité il est alors très intéressant de la combiner avec la méthode d'énumération par séparation et évaluation, ceci peut se faire en utilisant les résultats du lemme 4. Puisque les q_i sont des probabilités (donc pas des entiers) l'élagage suivant la première coordonnée est très probablement non linéaire (car il nécessite un tri de l'ordre de $|S^{(i)}|log|S^{(i)}|$, or $|S^{(i)}|$ peut être très grand), on se contentera d'un élagage partiel. Ceci nous donnera une famille d'algorithmes polynomiaux dont la complexité croît avec l'amélioration de l'optimalité de la solution, une telle famille est appelée un schéma d'approximation (voir [28]).

Le schéma d'approximation qu'on va présenter est basé sur des modifications dans l'élagage de l'algorithme exact APDPOTP : on ordonne les $S^{(i)}$ suivant l'ordre croissant des coûts des ordonnancements partiels (c'est à dire suivant la première coordonnée plutôt que la seconde). Le tri est très probablement non linéaire ainsi on propose de faire un tri par intervalle (c'est à dire on groupe les couples ayant la première coordonnée dans le même sous intervalle). D'après le lemme 4 on retient seulement un couple. Ainsi on a l'élagage suivant :
(élagage de $S^{(i)}$) **Pour** tous les triplets $(T_1, t_1, b_1), (T_2, t_2, b_2) \in S^{(i)}$ *et* T_1, T_2 dans le même intervalle et $t_1 \leq t_2$ alors $S^{(i)} = S^{(i)} - \{(T_1, t_1, b_1)\}$

Des bornes de la solution optimale nous donnent la taille de l'intervalle à subdiviser : soit $C_1 = \sum_{i=1}^{n} q_i \sum_{j=1}^{i} q_j p_j$ le coût optimal quand $m = 1$ on a alors (voir [20])

$$\frac{m+n}{m(n+1)} C_1 \leq wmft_{P_1}(POTP) \leq C_1$$

Donc pour $m = 2$ $wmft(POTP) \in [C_1/2, C_1]$. Divisons l'intervalle $[0, C_1]$ en $\lceil 2n/\varepsilon \rceil$ sous intervalles de taille $\lceil \varepsilon C_1/(2n) \rceil$. Les règles d'élagage retiennent seulement le meilleur uplet dans chaque sous intervalle. L'erreur introduite à chaque étape est au plus $\lfloor \varepsilon C_1/(2n) \rfloor$, donc l'erreur totale ne dépasse pas $\varepsilon C_1/2$. Chaque $S^{(i)}$ à au plus $\lfloor (2n)/\varepsilon \rfloor$ uplets donc la complexité est $O(n^2/\varepsilon)$. Cette méthode est appelée subdivision statique.

Remarques :

Les algorithmes ci dessus ne donnent une bonne solution qu'au prix d'une énumération de plus en plus grande de l'espace des solutions.
Une autre méthode est proposé dans [58] appelée subdivision dynamique : soit $T_i(S^{(i)})$ la plus grande valeur du coût dans $S^{(i)}$ (on calculera $T_i(S^{(i)})$ avant d'éliminer les éléments appartenant à un même intervalle). On partage à chaque étape i l'intervalle $[0, T_i(S^{(i)})]$ en $\lceil 2n/\varepsilon \rceil$ sous-intervalles et on retient pour chacun de ces intervalles un élément de $S^{(i)}$. La subdivision statique donne généralement une meilleur solution que la subdivision dynamique. Une autre approche intéressante serait la combinaison des deux subdivisions : on commence par une subdivision statique jusqu'à ce que la plus petite valeur dans $S^{(i)}$ soit plus grande que $C_1/2$, puis on utilise une subdivision dynamique. Cette méthode combinée donne une meilleur solution que les deux subdivisions statique et dynamique [58].

Procédures *Baker* et *Merteen* ont proposé des heuristiques de liste pour les problèmes de temps d'achèvement pondéré. Ces heuristiques sont fondés sur les résultats des problèmes non pondéré. Malheureusement on a presque pas de résultat théorique de leurs efficacité mais l'étude expérimentale donne des résultats satisfaisants.

Deux approches sont données : pour les deux approches on sélectionne un ordre a priori ensuite pour la première approche on affecte les tâches une par une, on note H_1 cette stratégie et H_m celle qui consiste à affecter m tâches à la fois.

H_1 :

- Sélectionner un ordre de priorité pour les tâches.
- Répéter jusqu'à ce que toutes les tâches soient affectées. Affecter la tâche la plus prioritaire à la machine la moins engagée.
- Ordonner les tâches suivant $WSPT$ dans chaque machine.

H_m :

- Sélectionner un ordre de priorité pour les tâches.
- Répéter jusqu'à ce que toutes les tâches soient affectées. Affecter les m tâches les plus prioritaires aux m machines sachant que la tâche ayant la plus grande pondération est affectée à la machine la moins engagée.
- Ordonner les tâches dans chaque machine suivant $WSPT$.

Remarques

1. Dans le cas où les pondérations sont égales les deux algorithmes donnent la solution optimale pour l'ordre SPT, ce résultat a motivé l'étude des deux heuristiques en parallèle

2. Pour chaque règle de priorité on a une heuristique. Il y a au moins cinq règles intéressantes : SPT (*Shortest Processing Time*), LPT (*Largest Processing Time*), WSPT, WLPT (*weighted Largest processing time*) et W plus grande pondération.

3. Le choix des règles pour n petit n'est pas clair. Pour chaque règle on peut trouver un problème où cette règle est la meilleure. Par contre pour n grand, l'une des deux règles $WSPT$, $WLPT$ donne le meilleur ordonnancement.

4. La performance des heuristiques dépend de la distribution des temps d'exécution, et du nombre de machines.

5. La stratégie $H_1(WSPT)$ est la procédure recommandée pour plusieurs raisons : elle est la plus facile à implémenter, elle donne très probablement la meilleure solution, elle est plus rapide (on n' a pas besoin de réordonner), c'est la règle optimale quand les probabilités sont égales. $H_1(WSPT)$ est une heuristique en pire cas constant d'après le paragraphe 3.6.1

3.6 Stratégie de Réaffectation

3.6.1 Stratégie de Réaffectation à plusieurs Machines Identiques

On suppose que les machines sont identiques, alors la solution optimale [11] consiste à ranger les tâches en une file par ordre croissant de service et de distribuer au fur et à mesure sur les machines (i.e, si l'ordonnancement total est (1,2,3,..n) et si on a m machines, alors la machine j doit traiter dans l'ordre les tâches (j,j+m,j+2m,...). La stratégie de réaffectation qu'on note Σ_{Reafct} consisterait alors à réaffecter au fur et à mesure les tâches présentes aux machines suivant l'ordre total.

Espérance de la Stratégie de Réaffectation à plusieurs Machines Identiques

Théorème 13 *Soient n le nombre de tâches et m le nombre de machines, une tâche i a une probabilité de présence q_i, les machines sont identiques et soit $(1,2,..n)$ l'ordonnancement total. on a alors :*

$$\mathbb{E}(\Sigma_{Reafct}) = \sum_{i=1}^{n} p_i q_i \left[1 + \sum_{k=1}^{\lfloor \frac{n-i}{m} \rfloor} \sum_{j=i+mk}^{n} q_j \mathbb{P}(mk-1 \text{ tâches présentes parmi } (i+1,....,j-1)) \right] \tag{3.23}$$

Démonstration : La durée p_i apparaît dans la fonction objective si et seulement si la tâche i est présente (avec une probabilité q_i) ; dans ce cas elle apparaît en plus chaque fois qu'une tâche $j > i$ est présente et la tâche j est affectée à la même machine que i, i.e que $mk-1$ tâches présentes parmi $(i+1,...,j-1)$ pour $k \in \{1,...\lfloor \frac{n-i}{m} \rfloor\}$. ∎

Remarques :

- Si les tâches ont la même probabilité q alors

$$\mathbb{E}(\Sigma_{Reafct}) = \sum_{i=1}^{n} p_i q [1 + q \sum_{k=1}^{\lfloor \frac{n-i}{m} \rfloor} \sum_{j=0}^{n-mk-i} \binom{mk-1+j}{mk-1} (1-q)^j q^{km-1}]$$

- La stratégie de réaffectation est équivalente à la stratégie de réoptimisation : puisque le rangement n'est pas perturbé lorsqu'on supprime certaines tâches donc pour chaque sous ensembles S la réaffectation donne la solution optimale.

Complexité de la Stratégie de Réaffectation à plusieurs Machines Identiques

On va montrer que l'espérance de Σ_{Reafct} se calcule en un temps polynomial.

Proposition 27 *La stratégie de réaffectation se calcule en un temps polynomial.*

Démonstration :
On considère la propriété suivante :
$\mathcal{P}_n$:Pr(s présent parmi {2,...n }) se calcule en un temps polynomial $\forall$ s $0 \leq s \leq n-1$.
On va montrer que $\mathcal{P}_n$ est vraie pour tout n par récurrence sur n.

- Pour n=2 le résultat est vrai en effet :

$$\begin{aligned} \mathbb{P}(s \text{ présent } parmi \ \{2\}) &= 1 - q_2 \ \ si \ \ s = 0 \\ &= q_2 \ \ sinon \end{aligned}$$

- pour n deux cas se présentent

  - $0 \leq s \leq n-2$, on a le résultat d'après l'équation suivante et l'hypothèse de récurrence $\mathcal{P}_{n-1}$

$$\begin{aligned} \mathbb{P}(s \text{ présent } parmi \ \{2,...n\}) &= q_n \mathbb{P}(s-1 \text{ présent } parmi \ \{2,....,n-1\}) \\ &+ (1-q_n)\mathbb{P}(s \text{ présent } parmi \ \{2,...,n-1\}) \end{aligned}$$

  - $s = n-1$

$$\mathbb{P}(n-1 \text{ présent } parmi \ \{2,...n\}) = \prod_{i=2}^{n} q_i$$

∎

Relation entre le POTP et la Stratégie de Réaffectation à plusieurs Machines Identiques

On note $\tilde{\mathbb{E}}(f) = \mathbb{E}(f) - \sum_{i=1}^{n} p_i q_i$. Supposons que n est un multiple de m (sinon on ajoute des tâches de temps d'exécution nul). On a alors le résultat suivant :

Proposition 28 *Soient n le nombre de tâche, m le nombre de machines, les tâches sont présentes avec une même probabilité q.*

$$\tilde{I\!E}(\Sigma_{Reafct}) \geq sup(q^{n-m-1}, q^{m-1}(1-q)^{n-m-1})\tilde{I\!E}(mft(POTP)$$

Démonstration : Puisque n est un multiple de m, alors

$$\tilde{\mathbb{E}}(mft(POTP)) = q^2 \sum_{i=1}^{n-m} \lfloor \frac{n-i}{m} \rfloor p_i$$

Dans le cas ou toutes les tâches ont la même probabilité de présence on a d'après (3.23) :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n} p_i q^2 \sum_{k=1}^{\lfloor \frac{n-i}{m} \rfloor} \sum_{j=0}^{n-mk-i} \binom{mk-1+j}{mk-1} (1-q)^j q^{km-1}$$

- On va montrer

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) \geq q^{m-1}(1-q)^{n-m-1} \tilde{\mathbb{E}}(mft(POTP) \tag{3.24}$$

On a

$$(1-q)^{n-m-1} \sum_{j=0}^{n-m-i} \binom{m-1+j}{m-1} \leq \sum_{j=0}^{n-m-i} \binom{m-1+j}{m-1} (1-q)^j$$

Or

$$\frac{n-i}{m} \leq \sum_{j=0}^{n-m-i} \binom{m-1+j}{m-1} = \binom{n-i}{m}$$

donc

$$\lfloor \frac{n-i}{m} \rfloor (1-q)^{n-m-1} \leq \sum_{j=0}^{n-m-i} \binom{m-1+j}{m-1} (1-q)^j$$

en sommant sur i on obtient :

$$q^{m-1}(1-q)^{n-m-1} q^2 \sum_{i=1}^{n-m} p_i \lfloor \frac{n-i}{m} \rfloor \leq \tilde{\mathbb{E}}(\Sigma_{Reafct})$$

ce qui prouve 3.24.

- De même on a

$$q^{n-m-1} \lfloor \frac{n-i}{m} \rfloor \leq \sum_{k=1}^{\lfloor \frac{n-i}{m} \rfloor} q^{km-1}$$

d'ou la deuxième inégalité

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) \geq q^{n-m-1} \tilde{\mathbb{E}}(mft(POTP)) \tag{3.25}$$

D'après (3.24) et (3.25) on a la proposition.

∎

Remarques :

- On remarque que les bornes ne sont pas atteintes, en effet les inégalités sont strictes dès que $q < 1$.
- Les bornes ci dessus sont mauvaises. Pour m constant et n très grand la borne tend vers zéro, or les deux stratégies sont proches car les deuxième termes des égalités suivantes sont très petits pour n grand :

$$\sum_{j=0}^{n-mk-i} \binom{mk-1+j}{mk-1}(1-q)^j q^{km-1} = 1 - \sum_{j=n-mk-i+1}^{\infty} \binom{mk-1+j}{mk-1}(1-q)^j q^{km-1}$$

donc

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \tilde{\mathbb{E}}(mft(POT)) - q^2 \sum_{i=1}^{n-m} p_i \sum_{k=1}^{\lfloor \frac{n-i}{m} \rfloor} \sum_{j=n-mk-i+1}^{\infty} \binom{m-1+j}{m-1}(1-q)^j q^{km-1}$$

- On va écrire notre fonctionnelle sous une autre forme :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n-m} p_i q_i \sum_{j=i+1}^{n} \mathbb{P}(j \in M_i)$$

M_i étant la machine où la tâche i est exécutée. On pose

$$Y_j = \begin{cases} 1 & si\ j \in M_i \\ 0 & sinon \end{cases}$$

Soit N_i le nombre de tâches présentes après i sur la machine M_i alors $N_i = \sum_{j=i+1}^{n} Y_j$.

On a alors :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n-m} p_i q_i \mathbb{E}(N_i) \tag{3.26}$$

Si on note W_i le nombre de tâches présentes après i, alors $N_i = \lfloor \frac{W_i}{m} \rfloor$ et donc

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n-m} p_i q_i \mathbb{E}(\lfloor \frac{W_i}{m} \rfloor) \tag{3.27}$$

- L'expression (3.27) permet d'obtenir une nouvelle borne pour Σ_{Reafct}.

Proposition 29 *Soient n le nombre de tâches et m le nombre de machines, les tâches sont présentes avec une même probabilité q.*

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) \leq \tilde{\mathbb{E}}(mft(POTP)) \leq \tilde{\mathbb{E}}(\Sigma_{Reafct}) + q \sum_{i=1}^{n-m} p_i$$

Démonstration :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n-m} p_i q_i \mathbb{E}(\lfloor \frac{W_i}{m} \rfloor)$$

en utilisant le fait que $x - 1 \leq \lfloor x \rfloor \leq x$ et que la variable aléatoire W_i qui représente le nombre de tâche présente après i est une binômiale de paramètre $(n-i, q)$ on a :

$$\lfloor \frac{n-i}{m} \rfloor q - 1 \leq \mathbb{E}(\lfloor \frac{W_i}{m} \rfloor)$$

En multipliant par qp_i et sommant sur i on obtient le résultat ∎

Remarque : On remarque que si $p_1 \leq p_2 ..... \leq p_n$ et $q_1 \leq q_2 ..... \leq q_n$ alors :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) \leq \tilde{\mathbb{E}}(mft(POTP)) \leq \tilde{\mathbb{E}}(\Sigma_{Reafct}) + \sum_{i=1}^{n-m} q_i p_i$$

Proposition 30 *Soient n le nombre de tâches et m le nombre de machines, la tâche i est présente avec une probabilité* q_i.

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) \leq \tilde{\mathbb{E}}(mft(POTP)) \leq m\tilde{\mathbb{E}}(\Sigma_{Reafct}) + m \sum_{i=1}^{n-m} q_i p_i$$

Démonstration :

$$\tilde{\mathbb{E}}(\Sigma_{Reafct}) = \sum_{i=1}^{n-m} p_i q_i \mathbb{E}(\lfloor \frac{W_i}{m} \rfloor)$$

On pose

$$X_j = \begin{cases} 1 & si\ j\ est\ \text{présent} \\ 0 & sinon \end{cases}$$

Alors $W_i = \sum_{j=i+1}^{n} X_j$ on obtient :

$$\sum_{j=i+1}^{n} q_j - 1 \leq \mathbb{E}(\lfloor \frac{W_i}{m} \rfloor)$$

En multipliant par $q_i p_i$ et sommant sur i on obtient $\sum_{i=1}^{n-m} q_i p_i \sum_{j=i+1}^{n} q_j$ qui est le coût sur une seule machine :

$$\frac{1}{m}\tilde{\mathbb{E}}(mft(POTP)) - \sum_{i=1}^{n-m} q_i p_i \leq \sum_{i=1}^{n-m} q_i p_i \sum_{j=i+1}^{n} q_j - \sum_{i=1}^{n-m} q_i p_i$$

∎

3.6.2 Stratégie de Réaffectation à plusieurs Machines de Vitesse Différentes

On sait que dans le cas où les machines sont différentes la solution optimale est déterminée par l'algorithme de Ford et Fulkerson. Le cas où les machines sont identiques sauf en leurs vitesses relatives, on trouve (cf [17]) un algorithme plus simple pour résoudre ce nouveau problème basé sur le lemme suivant.

Lemme 5 *[17] Soit $a_1 \geq a_2 \geq .... \geq a_n$ et $b_1, ..., b_n$ deux suites de nombres. Une permutation σ minimise la quantité suivante :*

$$\sum_{i=1}^{n} a_i b_{\sigma(i)}$$

si et seulement si $b_{\sigma(1)} \leq .... \leq b_{\sigma(n)}$.

Pour modéliser ce problème, on va associer à la $i^{\text{ème}}$ machine un nombre positif $\gamma_i \geq 1$ (vitesse), si le temps d'exécution de la tâche k est p_k alors le temps d'exécution de la tâche k sur la machine i est : $p_{ik} = \gamma_i p_k$. On considère la matrice Γ suivante :

$$\Gamma = \begin{bmatrix} \gamma_1 & 2\gamma_1 & 3\gamma_1 & \dots & n\gamma_1 \\ \gamma_2 & 2\gamma_2 & 3\gamma_2 & \dots & n\gamma_2 \\ \vdots & & & & \vdots \\ \gamma_m & 2\gamma_m & 3\gamma_m & \dots & n\gamma_m \end{bmatrix}$$

On note γ_{ij} le $(i,j)^{\text{ème}}$ élément de Γ et soit

$$\gamma_{i_1 j_1}, \cdots, \gamma_{i_{m'} j_{m'}}$$

la suite d'élément de Γ ($m' = mn$) tel que

$$\gamma_{i_1 j_1} \leq \cdots \leq \gamma_{i_{m'} j_{m'}} \tag{3.28}$$

On remarque que tout ordonnancement peut être interprété comme une affectation des p_j avec les éléments de Γ. On associe p_k avec γ_{ij} pour $k = 1, ..., n$ si la tâche k est affectée à la $i^{\text{ème}}$ machine et si $j-1$ tâches sont affectées à la même machine après k. D'après le lemme 5 si on suppose que

$$p_n \leq \cdots \leq p_1 \tag{3.29}$$

alors la solution optimale consiste à associer p_k avec $\gamma_{i_k j_k}$. En utilisant un algorithme efficace d'implémentation de liste prioritaire, on peut construire l'ordonnancement optimal en un temps $O(nlogn + nlogm + m)$. Le $O(nlogn)$ est le temps correspondant au tri des temps d'exécution et $O(nlogm + m)$ correspond à la recherche de la liste prioritaire des termes $\gamma_{i_1 j_1} \leq \cdots \leq \gamma_{i_n j_n}$.

On va proposer un calcul explicite de la stratégie de réaffectation qui consiste à affecter les tâches présentes d'une façon optimale pour chaque exemplaire S.

Théorème 14 *Soient n le nombre de tâches et m le nombre de machines, une tâche i a une probabilité de présence q_i, les machines sont de vitesses différentes. On a alors :*

$$I\!E(\Sigma_{Reafct}) = \sum_{k=1}^{n} \gamma_{i_k j_k} \sum_{i=1}^{n-k+1} p_i q_i I\!P(k-1 \text{ tâches présentes parmi } (i+1, ..., n)) \tag{3.30}$$

Démonstration : $\gamma_{i_k j_k} p_i$ apparaît dans la fonction objective si et seulement si la tâche i est présente (avec une probabilité q_i) et si $k-1$ tâches plus prioritaires que i (c'est à dire parmi $(i+1,...,n)$) sont présentes .

En effet l'absence des tâches ne perturbe pas les inégalités 3.29, soit $S = \{s_1 \cdots s_k\}$ ($|S| = k$) l'ensemble des tâches présentes, d'après 3.29 on a :

$$p_{s_k} \leq \cdots \leq p_{s_1} \tag{3.31}$$

La liste prioritaire restreinte au k tâches est celle formée par les k premiers termes de la liste donnée par les inégalités 3.28, donc la solution optimale à travers S est obtenu en associant p_{s_k} à $\gamma_{i_k j_k}$. ∎

Remarques :

- Si les tâches ont la même probabilité de présence q alors

$$\mathbb{E}(\Sigma_{Reafct}) = \sum_{k=1}^{n} \gamma_{i_k j_k} \sum_{i=1}^{n-k+1} p_i \binom{n-i}{k} q^k (1-q)^{n-i}$$

- L'espérance de Σ_{Reafct} pour des machines à vitesse différentesse calcule en un temps polynomial d'après la propriété $\mathcal{P}_n$ de la Proposition 27.

3.6.3 Stratégie de Réaffectation à plusieurs Machines Différentes

On suppose dans ce paragraphe que les machines sont différentes alors d'après [11] le problème se ramène à un problème de flot maximal de coût minimal. On peut construire un graphe $G = (X, C)$ qui a la propriété d'admettre un flot maximal de coût minimal si et seulement si il existe un ordonnancement de coût minimal. Si la tâche i est suivie de s tâches sur la machine k sa durée contribue par la quantité $(s+1)p_i^k$. On considère alors la matrice à n colonnes et $m' = nm$ lignes définie par

$$C_{li} = (s+1)p_i^k \quad si \quad l = sm + k \quad 0 \leq s \leq n-1 \quad 1 \leq k \leq m \tag{3.32}$$

On sait que dans ce cas (machines différentes) la solution optimale est déterminée par l'algorithme de Ford et Fulkerson. On considère la stratégie de réaffectation qui consiste à affecter les tâches présentes d'une façon optimale pour chaque exemplaire S. On note Σ_{Reafct} cette stratégie.

L'ordre d'affectation des tâches est perturbé, en effet supposons que pour l'exemplaire S on a un ordre donné si on perturbe S par l'absence d'une de ses tâches notée i_0 alors si la tâche i_0 est suivie par d'autres tâches dans l'ordre initial de S le sommet l_{i_0} ($C_{l_{i_0} i_0}$ est la contribution de la tâche i_0 dans la solution optimale à travers S) n'est plus saturé donc l'ordre après i_0 peut être perturbé s'il existe une tâche j après i_0 tel que $C_{l_{i_0} j}$ est la plus petite contribution.

Remarque : Très probablement le calcul de la stratégie de réaffectation pour des machines différentes ne se fait pas en temps polynomial.

3.7 Stratégie de Réaffectation avec des Temps de Service Aléatoires

On se propose dans ce paragraphe d'analyser le problème d'Ordonnancement des Travaux avec des temps de service aléatoires X_i pour les tâches $i = 1...n$, les X_i étant indépendantes, à distribution exponentielle avec des paramètres différents λ_i. Et on suppose que chaque tâche i a une probabilité de présence q_i,

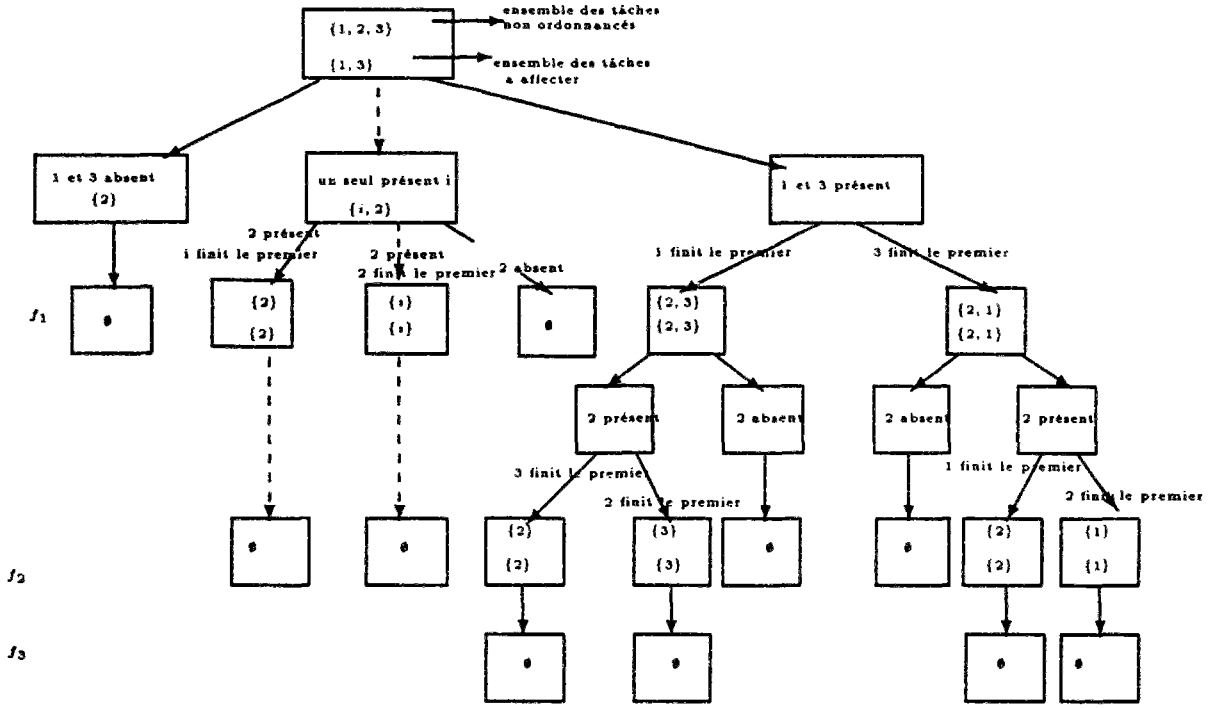
On va chercher une règle optimale pour la stratégie de réaffectation avec des temps de service aléatoires. Si V est l'ensemble des tâches alors on note $\mathbb{E}(\Sigma_{Reafct}(\pi, V, X_i, q_i))$ l'espérance au sens de la stratégie de réoptimisation quand la règle π est appliquée. $\mathbb{E}(\Sigma_{Reafct}(\pi, V, X_i, q_i))$ est une variable aléatoire par rapport aux temps de service. Le problème est de minimiser l'espérance de cette variable, (pour éviter toute confusion on notera cette dernière espérance par $\mathbf{Eu}(.)$) :

$$\min_{\pi} \mathbf{Eu}(\mathbb{E}(\Sigma_{Reafct}(\pi, V, X_i, q_i)) \tag{3.33}$$

La règle consiste à décrire l'ensemble des tâches à affecter aux machines à tout instant. Les règles de priorités sont d'un intérêt spécial : les tâches sont arrangées dans un ordre linéaire, et, à n'importe quel moment de décision, on affecte le plus possible de tâches non ordonnancées dans un ordre de priorité décroissant. En fait, pour minimiser la fonctionnelle coût en espérance il suffit de se restreindre à l'ensemble des règles qui ont des points de décision à l'instant zéro et au temps d'achèvement des tâches (voir [55]).

Exemple :

Supposons qu'on a deux machines et trois tâches 1,2 et 3, présentes avec une même probabilité q et des temps de service aléatoires X_i $i = 1...3$. L'ordonnancement commence à l'instant zéro, on sélectionne deux tâches pour l'exécution. Supposons qu'on affecte les machines aux tâches 1 et 3, d'après la propriété de la distribution exponentielle le prochain point de décision est l'instant $\frac{q^2}{\lambda_1+\lambda_3} + q^2(1-q)[\frac{1}{\lambda_1+\lambda_2} + \frac{1}{\lambda_3+\lambda_2}] + q(1-q)^2[\frac{1}{\lambda_1} + \frac{1}{\lambda_3} + \frac{1}{\lambda_2}]$. Dans le cas où 1 est la première achevée on affecte les machines aux tâches inachevées (c'est à dire $\{2,3\}$ si 3 est présent et 2 si 3 est absent) sinon on affecte les machines aux tâches $\{1,2\}$ si 1 est présent et à 2 si 1 est absent.



Ainsi on détermine notre ordonnancement c'est-à-dire qu'à n'importe quel point de décision on connaît l'ensemble de tâche à affecter. On note f_i la variable aléatoire qui désigne le $i^{\text{ème}}$ temps d'achèvement au $i^{\text{ème}}+1$ point de décision alors notre fonctionnelle est définie par $\mathbf{Eu}(\mathbb{E}(f_1)) + \mathbf{Eu}(\mathbb{E}(f_2)) + \mathbf{Eu}(\mathbb{E}(f_3))$. Il est clair que

$$\mathbf{Eu}(\mathbb{E}(f_1)) = \frac{q^2}{\lambda_1+\lambda_3} + q^2(1-q)[\frac{1}{\lambda_1+\lambda_2} + \frac{1}{\lambda_3+\lambda_2}] + q(1-q)^2[\frac{1}{\lambda_1} + \frac{1}{\lambda_3} + \frac{1}{\lambda_2}].$$

pour calculer $\mathbf{Eu}(\mathbb{E}(f_2))$ on l'écrit sous la forme :

$$\mathbf{Eu}(\mathbb{E}(f_2)) = \mathbf{Eu}(\mathbb{E}(f_1)) + \mathbf{Eu}(\mathbb{E}(f_2 - f_1))$$

On conditionne le deuxième terme (par les événements 1 (respectivement 3) est le premier achevé et 1 ou 3 présent). On a alors

$$\begin{aligned}
\mathbf{Eu}(\mathbb{E}(f_2 - f_1)) &= \frac{q^2}{\lambda_1+\lambda_3}\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ et } 3 \text{ présent et } 1 \text{ est le premier achevé}) \\
&+ \frac{q^2}{\lambda_1+\lambda_3}\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ et } 3 \text{ présent et } 3 \text{ est le premier achevé}) \\
&+ q(1-q)\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ absent } 3 \text{ présent}) \\
&+ q(1-q)\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|3 \text{ absent } 1 \text{ présent}) \\
&+ (1-q)^2\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ et } 3 \text{ absent })
\end{aligned}$$

comme

$$\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ et } 3 \text{ présent et } 1 \text{ est le premier achevé}) = q\frac{1}{(\lambda_2+\lambda_3)} + (1-q)\frac{1}{\lambda_3}$$

$$\mathbf{Eu}(\mathbb{E}(f_2 - f_1)|1 \text{ et } 3 \text{ absent }) = 0$$

donc

$$\begin{aligned}
\mathbf{Eu}(\mathbb{E}(f_2)) &= \frac{q^2}{\lambda_1+\lambda_3} + q^2(1-q)[\frac{1}{\lambda_1+\lambda_2} + \frac{1}{\lambda_3+\lambda_2}] + q(1-q)^2[\frac{1}{\lambda_1} + \frac{1}{\lambda_3} + \frac{1}{\lambda_2}] \\
&+ \frac{q^2\lambda_1}{\lambda_1+\lambda_3}(q\frac{1}{(\lambda_2+\lambda_3)} + (1-q)\frac{1}{\lambda_3}) \\
&+ q^2(1-q)\frac{\lambda_3}{\lambda_1+\lambda_3}(q\frac{1}{(\lambda_2+\lambda_1)} + (1-q)\frac{1}{\lambda_1}) \\
&+ q^2(1-q)\frac{1}{(\lambda_2+\lambda_3)}(\frac{\lambda_3}{\lambda_2} + \frac{\lambda_2}{\lambda_3}) \\
&+ q^2(1-q)\frac{1}{(\lambda_2+\lambda_1)}(\frac{\lambda_1}{\lambda_2} + \frac{\lambda_2}{\lambda_1})
\end{aligned}$$

De la même manière on peut calculer $\mathbf{Eu}(\mathbb{E}(f_3))$.

Nous allons maintenant donner une formulation générale du problème. Supposons que les tâches $1,...,r$ sont affectées. Puisque les X_i sont à distribution exponentielle alors $min\{X_1,...,X_r\}$ est exponentielle avec un paramètre $\lambda = \sum_{i=1,}^{r} \lambda_i$, et la probabilité que la tâche i soit la première achevée est $Pr(min\{X_1,...,X_r\} = X_i) = \frac{\lambda_i}{\lambda}$, alors d'après la propriété de perte de mémoire et la définition de notre fonctionnelle on a la formule suivante :

$$\begin{aligned}
\mathbf{Eu}(\mathbb{E}(\Sigma_{Reafct}(\pi, V, X_i, q_i)) &= \sum_{S\subseteq\{1,..,n\}} p(S)\frac{|S|}{\lambda_{min,S}} \qquad (3.34) \\
&+ \sum_{S\subseteq\{1,..,n\}} p(S) \sum_{k \text{ affectée}, k\in S} \frac{\lambda_k}{\lambda_{min,S}} \mathbf{Eu}(\Sigma_{Reafct}(\pi, S-\{k\}, X_i, q_i))
\end{aligned}$$

$\lambda_{min,S}$ est le paramètre de $X_{min,S}$ = minimum des temps d'exécution des tâches affectées quand S est le sous ensemble des tâches présentes.

Minimisation du délai d'exécution non pondéré

Dans le cas où toutes les tâches sont présentes ($q = 1$) on a d'après [26] le résultat suivant.

Théorème 15 *(Glazebrook) L'espérance de la somme des temps d'exécution dans le système est minimisée uniquement par la stratégie SEPT(Shortest expected processing time) où les tâches sont ordonnancées dans l'ordre croissant de leurs moyennes d'exécution.*

Proposition 31 *L'espérance de la moyenne de la somme des temps d'exécution dans le système (3.33) est minimisée uniquement par la stratégie SEPT.*

Démonstration : Par définition on a

$$\mathbf{Eu}(E[\Sigma_{Reafct}(\pi, V, X_i, q_i)]) = \sum_{S\subseteq V} p(S)\mathbf{Eu}[\Sigma_{Reafct}(\pi, S, X_i, q_i)]. \qquad (3.35)$$

D'après le théorème de *Glazebrook* $SEPT$ minimise $\mathbf{Eu}[\Sigma_{Reafct}(\pi, V, X_i, q_i)]$ pour $S = V$, or l'introduction de probabilité de présence pour les tâches ne perturbe pas la règle donc pour tout $S \subseteq V$, $SEPT$ minimise $\mathbf{Eu}[\Sigma_{Reafct}(\pi, S, X_i, q_i)]$. Comme le minimum d'une somme fini d'éléments positifs non liés est la somme des minimums donc (3.35) est minimisée uniquement par la règle $SEPT$. ∎

Minimisation du délai d'exécution pondéré

On trouve une analyse probabiliste dans le cas où toutes les tâches sont présentes ($q = 1$) pour le problème de la somme pondérée des temps d'exécution dans le papier de *Thomas Kampe* [45]. D'après *G Weiss et M Pinedo* [61] on a la proposition suivante qui se démontre par les même arguments invoqués pour la proposition précédente :

Proposition 32 *Si les poids w_i vérifient :*

$$\lambda_k w_l < \lambda_l w_k \rightarrow w_k \leq w_l$$

Le Σ_{Reafct} pondéré (3.33) est minimisé par la stratégie SEPT.

SEPT reste la règle de priorité optimale si les pondérations sont ordonnées de la même façon que les λ_i. Mais sous d'autres conditions on a :

Proposition 33 *Soit Π une règle de priorité statique $1 < 2.... < n$ où la tâche 1 a la plus grande priorité et n a la plus petite priorité. Si*

$$\begin{aligned} \lambda_1 w_1 &\geq ... \geq \lambda_n w_n \\ w_1 &\geq ... \geq w_n \end{aligned}$$

alors Π est optimal pour le Σ_{Reafct} pondéré (3.33).

Cette proposition est une conséquence directe du théorème 3 de *Thomas Kampe*[45].

Remarques :

- Pour n'importe quelle règle de priorité statique Π on peut trouver des pondérations w_i tel que Π est optimale.
- On remarque que bien qu'on connaît la règle optimale, il n'apparaît pas facile de calculer l'espérance pour un problème donné (même dans le cas où $q = 1$).
- la fonction coût est markovienne ; en effet c'est une somme de fonctions markoviennes.
- Il y a d'autres études probabilistes pour les POTs aléatoires. Par exemple *Cheng Nelson* et *Pinedo* (voir [14]) donnent un exemple où la règle optimale n'est même pas statique. Ils montrent dans le cas où on a seulement deux classes de tâche c'est-à-dire que les temps de service des tâches est soit μ_1 ou μ_2 et chaque tâche de classe i a une pondération w_i alors la règle optimale est dynamique, elle est du type seuil. En d'autres termes les décisions dépendent du nombre de tâches à exécuter dans chaque classe à l'instant t. Si $Q_1(t)$ et $Q_2(t)$ (le nombre de tâche dans la classe i à l'instant t) sont

plus grand que le seuil alors on applique $SEPT$, et elle est optimale. Sinon la règle $LEPT$(*Longest expected processing time*) qui affecte les m tâches avec le plus grand temps d'exécution est optimale. L'idée vient du fait que le système doit exécuter les tâches les plus importantes quand il y a un nombre suffisamment important, sinon il est plus judicieux d'équilibrer le chargement des machines quand le nombre des tâches est petit, pour plus de détails (*voir*[14]). On remarque que du fait que les décisions dépendant du nombre de tâches à exécuter dans chaque classe à l'instant t la règle qui minimise la fonctionnelle dans le cas déterministe (q=1) ne serait pas la même dans le cas probabiliste, donc pour chaque sous ensemble $S \subset V$ on aura une règle du type seuil.

3.8 Généralisations

3.8.1 POTP pondéré à une machine

On se propose d'étudier le cas où chaque tâche i est pondérée par w_i, donc le problème est d'ordonnancer le service des tâches afin de minimiser l'espérance de la moyenne pondérée des temps d'achèvement, alors on peut montrer la proposition suivante (par la même méthode que pour le cas où $w_i = 1 \ \forall i$) :

Proposition 34 *Soit n le nombre de tâches, on suppose que la tâche i a une probabilité de présence q_i et une pondération w_i. L'espérance du temps total pondéré passé dans le système de l'ordonnancement σ est :*

$$\mathbb{E}(wmft(\sigma)) = \sum_{i=1}^{n} q_{\sigma(i)} p_{\sigma(i)} \sum_{j>i} q_{\sigma(j)} w_{\sigma(j)} + \sum_{i=1}^{n} w_i q_i p_i. \tag{3.36}$$

les POTs pondérés sont stables dans le sens que si on perturbe le problème (en enlevant des tâches) il suffit de "gommer " les tâches absentes de l'ordonnancement optimal à travers toutes les tâches pour retrouver la solution optimale à travers les tâches présentes. Ceci est dû au fait que l'ordonnancement optimal à travers toutes les tâches est obtenue par la méthode gloutonne (ordonner les tâches par ordre croissant de priorité $\rho_i = \frac{w_i}{p_i}$), comme le rangement n'est pas perturbé par la modification, cette solution reste valable pour le POTP. Donc la "stratégie a priori" est équivalente à la "stratégie de réoptimisation".

3.8.2 POTP pondéré à plusieurs machines

Corollaire 8 *Soient n le nombre de tâches, m le nombre de machines, on suppose que la tâche i est présente avec une probabilité q_i. L'espérance du temps total pondéré passé dans le système pour l'ordonnancement $\xi = (\sigma_1, ..., \sigma_m)$ (où σ_i l'ordonnancement effectué par la $i^{ème}$ machine) est :*

$$\mathbb{E}(mft(\xi)) = \sum_{i=1}^{m} \mathbb{E}(wmft(\sigma_i)). \tag{3.37}$$

Remarques :

- Si les tâches ont la même probabilité de présence q alors $\mathbb{E}(mft(\xi))$ est égale à une constante près à $q^2 mft(\xi)$ d'où $POTP \equiv POT$.
- Le problème général se ramène à un problème de minimisation de délai d'exécution pondéré sur m machines sous les conditions suivantes :
  - la pondération $w_i' = w_i q_i \ \forall$ i
  - la durée de l'exécution de la tâche i est $q_i p_i \ \forall i \in T$.

3.8.3 Contraintes Potentielles

- **Contraintes de succession : un ordre partiel est imposé.**

  A. Espérance Une solution réalisable est une liste de m files d'attentes compatible avec l'ordre partiel. La modification $\mathcal{U}$ qui consiste à avancer dans la file (gommer les tâches absentes) donne une solution réalisable à travers les tâches présentes, car l'ordre sur l'ensemble des tâches présentes reste compatible avec l'ordre partiel.

  Le calcul de l'espérance d'une solution est indépendant de l'ordre partiel imposé, il se fait donc de la même façon que (3.1) et (3.9) on a alors :

$$\mathbb{E}(mft(\xi)) = \sum_{i=1}^{m} \mathbb{E}(mft(\sigma_i)).$$

$$\mathbb{E}(mft(\sigma_i)) = \sum_{k \in B_i} q_{\sigma_i(k)} p_{\sigma_i(k)} (1 + \sum_{j>k} q_{\sigma_i(j)}).$$

  L'ordre partiel entre les tâches est représenté par un graphe $G = (T, A)$ où T est l'ensemble des tâches et A l'ensemble des couples (i, j) tels que i précède j. Nous notons $G(S) = (T, A \cap S \times S)$ la restriction de l'ordre partiel à une partie S de T : soit ξ un ordonnancement des tâches T (un ordre sur T compatible avec l'ordre partiel) ; on note $\xi(S)$ la restriction de ξ à S en enlevant les tâches qui ne sont pas dans S.

  Si **un ordre partiel est imposé et** $m = 1$

  On rappelle la propriété $(\mathcal{P}1)$:

  $(\mathcal{P}1)$: si σ^* est la solution optimale du problème, soit S une partie de T alors $\sigma^*(S)$ est la solution optimale à travers S.

  La propriété $(\mathcal{P}1)$ est vérifiée quand un ordre partiel est imposé. Donc la stratégie de réoptimisation est équivalente à la stratégie a priori.

POTP	$\equiv$	POT
Stratégie a priori	$\equiv$	Stratégie de réoptimisation

  B. Complexité

  La complexité du problème est inconnue dans le cas général, on a pu déterminer des propriétés caractéristiques de la solution optimale. Ces propriétés ont permit [17] de

construire la solution optimale en un temps polynomial pour un ordre partiel particulier.

On sait que dans le cas où les tâches sont indépendantes, le rapport de priorité des tâches détermine la solution optimale. On peut généraliser la définition de rapport de priorité ρ_i à un sous-ensemble non vide [17].

Soit $S \subset T$ nous définissons $\rho(S)$ l'indice de priorité de la partie S :

$$\rho(S) = \frac{w(S)}{p(S)} \text{ où } p(S) = \sum_{i \in S} p_i, \quad w(S) = \sum_{i \in S} w_i$$

On va donner des propriétés caractéristiques de ces problèmes :

Nous dirons qu'une partie S est initiale si et seulement si :

(a) $S \neq \emptyset$

(b) tous les prédécesseurs de tâches de S sont dans S i.e si la tache j est avant la tâche i (selon l'ordre partiel), alors $j \notin S$ implique que $i \notin S$

On note $\mathcal{I}$ la classe des ensembles initiaux (respectivement à l'ordre partiel). On définit le nombre

$$\rho^* = \max_{S \in \mathcal{I}} \rho(S)$$

Nous dirons qu'une partie S est $\rho-$maximale si et seulement si :

(i) S est initiale.

(ii) $\rho(S)$ est maximal : $\rho(S) = \rho^*$.

(iii) S est minimal (pour l'inclusion) avec (i) et (ii) c'est à dire si $S_1 \subset \mathcal{I}$, $\rho(S_1) = \rho^*$ alors $S_1 = S$.

Ceci donne une caractérisation intéressante de l'ordonnancement optimal, et permet de proposer un algorithme efficace quand l'ordre est une famille *d'anti − arborescence*. Les parties $\rho-$maximales sont propriétaires dans l'ordonnancement [17]. En effet, on peut démontrer les 2 propriétés suivantes :

($\mathcal{P}2$) : Si S est une partie $\rho-$maximale, il existe une solution optimale σ telle que $\sigma = (\sigma(S), \sigma(\bar{S}))$.

($\mathcal{P}3$) : Si σ est une solution optimale, il existe une partie $\rho-$maximale S telle que $\sigma = (\sigma(S), \sigma(\bar{S}))$.

On ne connaît pas pour le moment de méthode efficace pour déterminer les parties $\rho-$maximale pour un ordre partiel quelconque. Pour le cas où l'ordre est une famille *d'anti − arborescence* (dans ce cas chaque tâche a au plus un successeur) un algorithme en $O(n^2)$ est proposé par *Bruno et al* [11]. Notons H_i la partie constituée de i et de ses prédécesseurs. On peut alors montrer que l'une des parties H_i où $i \in T$ est $\rho-$maximale. L' algorithme est construit comme suit :

Algorithme

(1) Détermination d'une tâche l telle que H_l soit ρ−maximale.

(2) D'après $\mathcal{P}2$, il existe une solution σ optimale telle que $\sigma = (\sigma(H_l), \sigma(\bar{H}_l))$: donc d'après la structure de l'ordre partiel G on a $\sigma = (\sigma(H_l - \{l\}), \{l\}, \sigma(\bar{H}_l))$. La tâche l est alors placée dans la permutation optimale.

(3) En remarquant que $G(H_l - \{l\})$ et $G(\bar{H}_l)$ sont aussi des familles d'anti-arborescences, et grâce à $\mathcal{P}1$. On peut réitérer les points (1) et (2) pour $G(H_l - \{l\})$ et $G(\bar{H}_l)$.

A chaque itération, une tâche supplémentaire est placée dans la permutation optimale. Le calcul des $\rho(H_l)$ nécessite O(n) opérations lorsque l'on balaye les sommets des sommets pendant vers la racine. Par conséquent, la complexité est bien $O(n^2)$.

- Dans le cas d'une arborescence l'algorithme se généralise en remplaçant les ensembles initiaux par des ensembles terminaux (tous les successeurs de S sont dans S) et les parties $\rho - maximale$ par les parties $\rho - minimale$ ($\rho^* = \min_{\text{S terminal}} \rho(S)$)
- Si les tâches sont indépendantes alors l'ensemble $\rho-$ maximal est la tâches de plus grande priorité.

- **Contraintes de localisation temporelle.**

Dans le cas où la tâche i doit être achevée à la date d_i ("due date"), alors tout ordonnancement admissible au sens POT reste admissible au sens POTP, car le temps d'achèvement des tâches diminue au cours des modifications.

On va étudier le cas où la tâche i ne peut commencer avant la date $r_i \neq 0$ ("release date"), on note $pred(j)$ la variable aléatoire qui représente la dernière tâche exécutée avant j. On montre la proposition suivante :

Proposition 35 *Soit n le nombre de tâches, une seule machine, on suppose que la tâche i ne peut commencer avant la date r_i et a une probabilité de présence q_i. L'espérance de l'ordonnancement $\sigma = (1, 2, ...n)$ est :*

$$\mathbb{E}(mft(\sigma)) = \sum_{i=1}^{n} q_i p_i (1 + \sum_{j>i} q_j \mathbb{P}(A_{i,j})) + \sum_{i=1}^{n} q_i r_i \mathbb{P}(D_i)(1 + \sum_{j>i} q_j \mathbb{P}(A_{i,j})). \quad (3.38)$$

$D_i = \{C_{pred(i)} \leq r_i\}$.

$A_{i,j} = \{p_i \text{ contribue dans } t_j\} = \bigcap_{k=i+1}^{j-1} A_{i,k} \cap D_j^c \cap \{pred(j) = k\}$ *pour* $j > i$.

Démonstration :

p_i apparaît dans la fonction objective si et seulement si i est présent et p_i réapparaît à chaque fois que p_i contribue dans t_j, pour tout j présente après i, or $t_j = max(r_j, C_{pred(j)})$, on note

$$A_{i,j} = \{p_i \text{ contribue dans } t_j\}$$

alors p_i contribue dans t_j si et seulement si p_i contribue dans $t_{pred(j)}$ et $max(r_j, C_{pred(j)}) = C_{pred(j)}$.

De même r_i apparaît dans le temps d'achèvement de la tâche i si et seulement si i est présent et $max(r_i, C_{pred(i)}) = r_i$, dans ce cas elle réapparaît autant de fois que p_i. ∎

Théorème 16 *La fonctionnelle ne se calcule pas en un temps polynomial.*

Démonstration : $\mathbb{P}(D_i)$ se calcule en $O(2^i)$ en effet :

$$\mathbb{P}(D_i) = \mathbb{P}(C_{pred(i)} \leq r_i)$$

on conditionne par $\{predi = k\}$, on a alors :

$$\mathbb{P}(D_i) = \sum_{k=1}^{i-1} \mathbb{P}(C_k \leq r_i)\mathbb{P}(pred(i) = k)$$

Or le temps de calcul de $\mathbb{P}(pred(i) = k) = q_k \prod_{l=k+1}^{i-1} (1 - q_l)$ est $i - k$. On note τ_k le temps de calcul de $\mathbb{P}(C_k \leq r_i)$ on a d'après les équations ci dessus

$$\tau_i = \sum_{k=1}^{i-1} \tau_k + \frac{i(i-1)}{2}$$

donc

$$\tau_i = 2\tau_{i-1} + 1$$

D'ou le résultat.

∎

Remarques :

- Dans le cas où les $r_i = 0$, on retrouve bien l'équation (3.1) puisque $D_i = \emptyset$ et $\mathbb{P}(A_{i,j}) = 1$.
- Le résultat de la Proposition 35 se généralise dans le cas de plusieurs machines. L'espérance de l'ordonnancement $\xi = (\sigma_1, .., \sigma_i, ...\sigma_m)$ est :

$$\mathbb{E}(mft(\xi)) = \sum_{i=1}^{m} \mathbb{E}(mft(\sigma_i)). \tag{3.39}$$

L'espérance de l'ordonnancement de σ_i est donné par l'équation 3.38.

- Le calcul de $\mathbb{P}(A_{i,j})$ se déduit de $\mathbb{P}(D_i)$, en effet puisque $A_{i,j}$ est la réunion d'événements disjoints $A_{i,pred(j)} \cap D_j \cap \{pred(j) = k\}$,

$$\mathbb{P}(A_{i,j}) = \sum_{k=i+1}^{j-1} \mathbb{P}(A_{i,pred(j)} \cap D_j \cap \{pred(j) = k\})$$

les événements $A_{i,pred(j)}$, D_j et { pred(j)=k } sont indépendants donc

$$\mathbb{P}(A_{i,j}) = \sum_{k=1}^{j-1} \mathbb{P}(\{pred(j) = k\})\mathbb{P}(A_{i,k})\mathbb{P}(D_k)$$

avec

$$\mathbb{P}(A_{i,i+1}) = \mathbb{P}(max(r_{i+1}, C_i) = C_i) = 1 - \mathbb{P}(D_{i+1})$$

- La variable aléatoire C_i s'écrit explicitement en fonction des temps d'achèvement des tâches qui précédent i :

  On pose

$$X_j = \begin{cases} 1 & si\ j\ est\ \text{présent} \\ 0 & sinon \end{cases}$$

donc $C_1 = p_1 + r_1$, $C_2 = p_2 + X_1 max(r_2, p_1 + r_1) + (1 - X_1)r_2$ et

$$C_i = p_i + X_{i-1} max(r_i, C_{i-1}) + \sum_{k=1}^{i-2} X_k \prod_{j=k+1}^{i-1} (1 - X_j) max(r_i, C_{i-1-k}) + \prod_{j=1}^{i-1} (1 - X_j) r_i$$

3.8.4 Contraintes de Ressources

Nous avons remarqué que les résultats du paragraphe 3.5 restent vrais dans le cas où les machines ne sont pas identiques, il suffit de changer p_i par p_i^k si la $i^{\text{ème}}$ tâche est affectée à la $k^{\text{ème}}$ machine. Pour le cas où les machines sont uniformes (c'est à dire $p_i^k = f_k p_i$) (*Horowit* et *Sahni* 1976) proposent une amélioration de leur approximation (proposée au paragraphe 3.5 en $O(n \log mn)$.

3.8.5 Fonction Economique

La durée totale de l'ordonnancement, notée $C_{max} = \max_i C_i$ fait parti des fonctions les plus utilisées dans le cas déterministe. Le problème est $NP - complet$ dès que $m = 2$(voir *Graham* [29] et [30]) une analyse probabiliste serait intéressante. Mais pour la méthode de modification $\mathcal{U}$ qui consiste à "gommer" dans chaque classes B_i, le calcul de la fonctionnelle peut demander un temps exponentiel. Cependant il y a des cas particuliers où on a pu donner la stratégie de réoptimisation.

- m=1

Dans ce cas tous les ordonnancements ont le même coût $Cmax = \sum_{i=1}^{n} p_i$ (au sens déterministe) on remarque que dans le cas où la tâche i est présente avec une probabilité q_i alors :

$$\mathbb{E}(C_{max}) = \sum_{i=1}^{n} q_i p_i \tag{3.40}$$

Comme toute les solutions ont le même coût alors :

POTP	≡	POT
Stratégie a priori	≡	Stratégie de réoptimisation

- $m \geq 1$

Dans le cas de plusieurs machines un ordonnancement $\xi = (\sigma_1, .., \sigma_i, ...\sigma_m)$ alors on note C^i_{max} la durée totale de l'ordonnancement σ_i donc :

$$Cmax = \max_i C^i_{max}.$$

On conditionne :

$$\mathbb{E}(Cmax) = \sum_{i=1}^{m} \mathbb{E}(Cmax = C^i_{max})\mathbb{P}(Cmax = C^i_{max}) \tag{3.41}$$

d'après (3.40)

$$\mathbb{E}(Cmax = C^i_{max}) = \sum_{k=1}^{|B_i|} p_{\sigma_i(k)} q_{\sigma_i(k)} \tag{3.42}$$

les variables aléatoires C^i_{max} étant indépendantes alors :

$$\mathbb{P}(Cmax = C^i_{max}) = \prod_{j=1}^{m} \mathbb{P}(C^j_{max} \leq C^i_{max}) \tag{3.43}$$

Théorème 17 *La fonctionnelle ne se calcule pas en un temps polynomial.*

Démonstration :

On note $\Omega(X)$ l'espace d'état de la v.a X alors :

$$Pr(C^j_{max} \leq C^i_{max}) = \sum_{x \in \Omega(C^i_{max})} Pr(C^j_{max} \leq x) Pr(C^i_{max} = x)$$

Comme

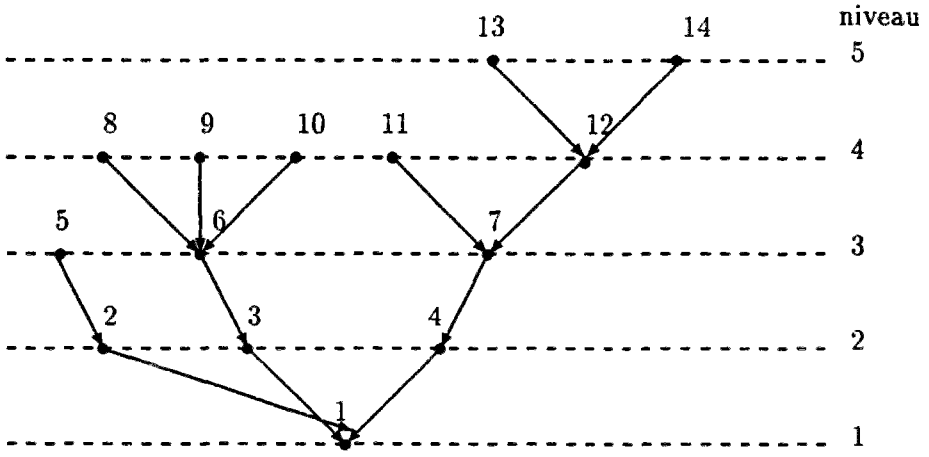
$$C^j_{max} = \sum_{k=1}^{|B_j|} p_{\sigma_i(k)} 1_{\{\sigma_i(k) \text{présent}\}}$$

Le calcul de $Pr(C^i_{max} \leq x)$ se fait en un temps $2^{|B_i|}$. Notre fonctionnelle se calcule en $0(2^{\max_i |B_i|})$ donc au moins $0(2^{\frac{n}{m}})$ puisque au meilleur des cas on a $\frac{n}{m}$ objets par machine (au pire des cas on a une tâche pour les $m-1$ machines et $n-m+1$ affectées à la dernière machine) .

∎

- Le graphe G est une *anti – arborescence* et les p_i sont tous égaux.

  Une notion qui paraît importante pour ce genre de problème est celle de niveau d'un sommet. c'est à dire la longueur du plus long chemin allant du sommet au sommet terminal.



puits

Graphe

Machine \ Temps	1	2	3	4	5	6
1	14	12	8	6	3	1
2	13	10	7	4		
3	11	9	5	2		

Solution Optimale

On construit une liste L ordonnée selon les niveaux non-croissants et l'on affecte les tâches prioritaires aux machines disponibles selon L : l'ordonnancement obtenu est optimal. la figure ci dessus rapporte les données d'un exemple ainsi que l'ordonnancement optimal sur trois machines pour $L = (14, 13, 12, ...1)$. On supposera que le puits 1 est toujours présent. Si on note N le nombre de niveau dans G alors $C_{max} = (N+1) * p$ si $p_i = p$. On a la proposition suivante :

Proposition 36 *Si le graphe G est une anti – arborescence, $p_i = p \ \forall \ i$ et le puits 1 présent alors*

$$I\!E(\Sigma_{Reafct}(C_{max})) = p \sum_{i=0}^{N} (i+1) Pr(\textit{niveau du graphe} = i). \tag{3.44}$$

Démonstration : Par définition on a :

$$\mathbb{E}(\Sigma_{reopt}(C_{max})) = \sum_{S\subseteq\{1,\ldots,n\}} \mathbb{P}(S)C_{max}(S).$$

le graphe G étant une $anti-arborescence$, 1 présent donc tout sous graphe de G est une $anti-arborescence$. Comme le rangement L n'est pas perturbé par la présence ou pas des tâches alors si S est de niveau $k-1$ l'ordonnancement optimal à travers S est : $C_{max}(S) = kp$.

∎

Le calcul de la fonctionnelle (3.44) se fait en un temps 2^N, en effet pour calculer $Pr(niveau\ du\ graphe = N-i)$ on a $\binom{N}{i}$ ensembles distincts donc si N est de l'ordre de n le calcul se fait en un temps exponentiel.

Dans le cas ou le nombre de tâche dans chaque niveau est le même, soit n_1 le nombre de tâche dans chaque niveau, on a alors.

$$\mathbb{E}(\Sigma_{reopt}(C_{max})) = p\sum_{i=0}^{N}(i+1)(1-(1-q)^{n_1})^i((1-q)^{n_1})^{N-i}.$$

3.9 Résumé

m	p_i^k	w_i	$<$	q_i	complexite		POTP
					POT	POTP	
—	p_i	egaux	$\emptyset$	egaux	$0(nlogn)$		POT
				—	$0(nlogn)$	NP-complet	—
—	—	egaux	$\emptyset$	egaux	$0(n^3)$		POT
				—	$0(n^3)$	NP-complet	—
1	—	—	arborescence	—	$0(n^2)$		POT
—	—	egaux	—	egaux	NP-complet		POT
				—			—
≥ 2	—	—	$\emptyset$	egaux	NP-complet		POT
				—			—

- m : nombre de machines.
- p_i^k : temps d'exécution de la tâche i sur la $k^{\text{ème}}$ machine.
- w_i : pondération de la tâche i.
- $<$: contraintes de succession entre les tâches.
- q_i : probabilité de présence de la tâche i.

3.10 Conclusion

L'etude dans ce chapitre est consacré à la recherche de cas particuliers où la nouvelle version probabiliste du problème est facile (P). Le résultat le plus général est celui obtenu dans le Paragraphe 3.4 pour les problèmes d'ordonnancement des travaux probabilistes à une machine : pour la première fois nous avons obtenu un problème qui reste polynomial quelque soit la probabilité. Dans le Paragraphe 3.5 nous avons aussi partiellement généralisé ce résultat au problème d'ordonnancement des travaux probabiliste à plusieurs machines dans le cas particulier où toutes les tâches ont la même probabilité de présence. Ainsi l'introduction des probabilités dans les POCs nous a permis d'analyser la stabilité des solutions de certains problèmes d'optimisation combinatoires quand ils sont perturbés par l'absence de certaines données.

L'étude des POTPs nous a permis de déduire une condition de stabilité de la solution du problème déterministe : les problèmes qui sont résolubles par un algorithme de liste imperturbable par l'absence de certaines données sont stables, nous remarquons qu'une telle propriété reste vraie pour les problèmes stables que nous avons proposé dans les Paragraphes 3.8.3 et 3.8.4.

Nous avons aussi obtenu une expression analytique explicite pour la fonctionnelle associée à la stratégie de réoptimisation du POT à plusieurs machines dans les paragraphes 3.6.1,3.6.2 et 3.6.3 . Ceci nous permet d'avoir une comparaison explicite des deux stratégies a priori et de réoptimisation. Nous montrons dans les paragraphes 3.6.1,3.6.2 que le calcul de la fonctionnelle de la stratégie de réoptimisation du POT se fait en temps polynomial, dans le cas de machines identiques ou uniformes. Nous obtenons ainsi le premier POCP dont le calcul de l'équation (0.2) se fait en temps polynomial.

Chapitre 4

Problèmes de Bin-Packing Probabilistes

Le Bin-Packing est caractérisée par la nécessité de placer une collection d'objets dans des régions bien définies ; afin de donner la meilleure utilisation du temps ou de l'espace. Ces problèmes apparaissent dans un grand nombre de domaines : l'informatique (gain de mémoire), gestion de production, etc.

Le Problème du Bin-Packing est NP-dur [25], mais il joue un rôle très important dans la théorie de la complexité car c'est un des rares POCs qui sont NP-dur et pour lequel il existe des heuristiques très performantes [44].

Deux approches analytiques sont proposés pour le Bin-Packing. L'approche statique (si on connaît a priori toutes les données le concernant), on analyse alors la performance des heuristiques en pire cas [42], [44] (la déviation maximale de la solution optimale). Plusieurs travaux ([52],[16]) ont développé une analyse probabiliste (la taille des objets est une variable aléatoire), ceci permet d'étudier le comportement asymptotique de la solution optimale et d'analyser la performance en moyenne des différentes heuristiques du Bin-Packing.

Le modèle que nous allons étudier dans ce chapitre consiste à supposer que la présence des objets est connue seulement en probabilité, (on obtient un problème plus adapté à la réalité). L'étude de ce modèle représente une généralisation des travaux ([9],[35],[36],[37]) sur les problèmes d'optimisation combinatoires probabilistes (POCPs). En effet ceci nous permet d'élargir les notions de stratégie a priori à des problèmes d'optimisation combinatoires non définis sur des graphes.

Nous donnons au Paragraphe suivant les définitions et les notations des Problèmes de Bin-Packing (PBP). Dans le Paragraphe 2 nous donnons la définition des Problèmes de Bin-Packing Probabilistes (PBPP), en particulier on définit les stratégies de redistribution suivant une heuristique. Dans le Paragraphe 3 nous discutons les motivations et certaines applications du PBPP. Nous commençons dans le Paragraphe 4.4 par proposer une stratégie sans redistribution pour le PBPP, en particulier nous proposons une expression de l'espérance de la fonctionnelle associée à cette stratégie, puis nous étudions la complexité du problème. Dans le Paragraphe 4.5 nous rappelons plusieurs heuristiques pour le PBP et la définition des stratégies de redistribution. Dans le Paragraphe 4.6 nous étudions la stratégie de redistribution suivant Next-Fit-Decreasing : nous proposons une expression de l'espérance de notre fonctionnelle et nous étudions la complexité du problème. Ensuite nous analysons les propriétés combinatoires de cette fonctionnelle, puis nous proposons une étude asymptotique

et probabiliste de cette approche. Dans le Paragraphe 4.7 nous introduisons la stratégie de redistribution suivant matching et nous donnons une expression explicite de l'espérance de la fonctionnelle associée. Enfin Dans le Paragraphe 4.8 nous montrons sous certaines conditions que la stratégie de ré-optimisation, et les stratégies de redistribution suivant Matching, Best-Fit-Decreasing et First-Fit-Decreasing sont asymptotiquement équivalentes.

4.1 Définitions et Notations Générales des Problèmes de Bin-Packing

Le modèle mathématique du Bin-Packing peut être défini ainsi : on dispose d' un nombre infini de boîtes de même capacité 1, et d'une liste de n objets qu'on note $L_n = (x_1, ..., x_n)$. On suppose que la taille des objets est inférieure à la capacité. Sans perte de généralité, on peut noter x_i le nom et la taille de l'objet.

Pour la suite de l'étude nous introduisons les notations suivantes :

a) $OPT(L_n) = OPT((x_1, ..., x_n))$ le nombre minimum de boîtes pour placer les objets $(x_1, ..., x_n)$

b) Un groupement (Packing) G des n objets est une partition des objets $(x_1, ..., x_n)$ en m boîtes $B_1, B_2, ..., B_m$. Sans perte de généralité, on peut poser : $B_i = \bigcup_{x_j \in B_i} \{x_j\}$.

c) On notera $|B_i|$ le nombre d'objets placés dans la boîte B_i.

d) Soit H une heuristique de Bin-Packing, on note $Q_n^H = Q_n^H(x_1, ..., x_n)$ le nombre de boîtes pour placer les objets $(x_1, ..., x_n)$ suivant H.

e) x_i est un $k - objet$ si $\frac{1}{k+1} < x_i \leq \frac{1}{k}$.

f) On appelle $k - boîte$ les boîtes qui commencent par un $k - objet$.

i) On note n_k le nombre de $k - objet$ dans la liste L_n $(\sum_{k \geq 1} n_k = n)$

Le PBP peut être énoncé ainsi :

PBP : Trouver le plus petit entier m tel qu' il existe une partition de la liste $L_n = B_1 \cup B_2, ... \cup B_m$ **et** $\sum_{x_i \in B_j} x_i \leq 1$ **pour tout j** $1 \leq j \leq m$.

4.2 Définition des Problèmes de Bin-Packing Probabilistes

La nouvelle approche du Bin Packing que nous allons étudier peut se modéliser de la façon suivante :

- Soit L_n une liste de n objets qu'on veut mettre dans des boîtes de capacité 1. On dispose d'une solution a priori G.

- On se donne une loi de probabilité $\mathbb{P}$ sur 2^{L_n}, l'ensemble de tous les sous-ensembles de L_n, i.e chaque exemplaire $S \subseteq L_n$ a une probabilité d'occurrence $\mathbb{P}(S)$.

  On se donne une méthode de modification $\mathcal{U}$: elle nous permet d'obtenir une solution à travers la liste S d'objets présents à partir de la solution a priori G. On note $G_S^{\mathcal{U}}$ la solution à travers S, et $Q_G^{\mathcal{U}}(S)$ le nombre de boîtes non vides dans $G_S^{\mathcal{U}}$, i.e :

  $$Q_G^{\mathcal{U}}(S) = \sum 1_{\{\text{boîte non vide} \in G_S^{\mathcal{U}}\}}$$

  Le problème du Bin-Packing probabiliste consiste à chercher une solution a priori G qui minimise l'espérance de $Q_G^{\mathcal{U}}$ (qui est donc une variable aléatoire sur 2^{L_n}).

$$\mathbb{E}(Q_G^{\mathcal{U}}) = \sum_{S \subseteq L_n} \mathbb{P}(S) Q_G^{\mathcal{U}}(S). \tag{4.1}$$

Nous proposerons deux méthodes de modification :

1. La stratégie Banale : $\mathcal{U}$ consiste à enlever les objets absents de chaque boîte de la solution initiale à travers les n objets (sans redistribution).
2. Stratégie de Redistribution suivant l'heuristique H : consiste à distribuer les objets présents suivant l'heuristique H pour chaque exemplaire S. On note Σ_{Redist}^{H} cette stratégie.

Il est clair que la stratégie qui distribue les objets présents d'une façon optimale pour chaque exemplaire est la stratégie optimale (on note cette stratégie Σ_{Redist}^{OPT} et on l'appelle stratégie de ré-optimisation) :

$$\mathbb{E}(\Sigma_{Redist}^{OPT}(L_n)) = \sum_{S \subseteq L_n} \mathbb{P}(S) OPT(S). \tag{4.2}$$

On note W (resp W_k) la variable aléatoire qui représente le nombre d'objets (resp $k - objet$) présents, et $W_k \, div \, k$, $W_k \, mod \, k$ t.q $W_k = k W_k \, div \, k + W_k \, mod \, k$.

4.3 Motivation du PBPP

Cette approche probabiliste pour le Bin Packing est intéressante du point de vue théorique. On sait que dans le cas où la taille x_j des n objets sont des variables aléatoires indépendantes et identiquement distribuées, le nombre de lots divisé par n tend presque sûrement vers une constante pour plusieurs heuristiques [52]. Des résultats similaires sont envisageables pour les fonctionnelles associées aux PBPP et à la stratégie de ré-optimisation : en effet nous démontrons dans le Paragraphe 4.6.6 que ce résultat reste vrai pour la stratégie de redistribution suivant Next Fit decreasing qu'on introduira ultérieurement. La constante est donnée explicitement, elle dépend de la constante de la fonctionnelle du problème déterministe. Dans le Paragraphe 4.7, on introduit la stratégie de redistribution suivant matching et nous montrons que c'est une stratégie asymptotiquement équivalente à la stratégie de ré-optimisation dans le Paragraphe 4.8. Ceci nous aidera à mieux comprendre le comportement asymptotique des stratégies a priori et de ré-optimisation pour un Problème d'Optimisation Combinatoire Probabiliste.

4.4 Stratégie Banale

La stratégie Banale consiste à enlever les objets absents de chaque boîte de la solution initiale à travers les n objets (sans redistribution). Le nombre de boîtes pour une solution donnée est alors une variable aléatoire, on va expliciter sa moyenne sous la stratégie Banale. On note $|G|$ la variable aléatoire qui représente le nombre de boîtes non vides pour une solution donnée G suivant la stratégie Banale.

Espérance d'une Solution au sens de la Stratégie Banale.

Proposition 37 *Soit n le nombre d'objets, on suppose que chaque objet i a une probabilité de présence p. L'espérance du nombre de boîtes sous la stratégie Banale d'un groupement* $G = B_1 \cup B_2, ..., \cup B_m$ *est :*

$$I\!E(|G|) = \sum_{i=1}^{m}(1 - (1-p)^{|B_j|}). \tag{4.3}$$

Démonstration : La boîte i contribue dans la fonction objective si et seulement si la boîte i n'est pas vide (probabilité $\mathbb{P}$($i^{\text{ème}}$ boîte vide)). La fonction objective est donc :

$$\mathbb{E}(|G|) = \sum_{i=1}^{m} \mathbb{P}(i^{\text{ème}} \text{ boîte non vide}). \tag{4.4}$$

On a :

$$\mathbb{P}(B_i \text{ vide }) = (1-p)^{|B_i|} \tag{4.5}$$

D'après (4.4) et (4.5) la fonction objective de la solution G est :

$$\mathbb{E}(|G|) = \sum_{i=1}^{m}(1 - (1-p)^{|B_j|}).$$

∎

Remarques :

- Si $|B_j| = k \ \forall j$ alors :

$$\mathbb{E}(|G|) = m(1 - (1-p)^k).$$

- Si on note G_{prob} la solution optimale de la stratégie banale, $G_{prob} = B_1 \cup B_2, ..., \cup B_m$, on pose $k(G_{prob}) = \inf_{1 \le i \le |G_{prob}|} |B_i|$, et soit G^* la solution optimale pour le Bin-packing à travers L_n :

$$\mathbb{E}(|G_{prob}|) \le \mathbb{E}(|G^*|) \le \mathbb{E}(|G_{prob}|)/(1 - (1-p)^{k(G_{prob})})$$

- Si les $|B_j|$ sont distincts le calcul de (4.3) se fait en $m * \sup_j |B_j|$. Donc si m est de l'ordre de n la fonctionnelle est en pire des cas en $0(n)$.

- Cette stratégie peut être utile si la redistribution est impossible : par exemple dans la situation où les boîtes ne communiquent pas.
- Cette stratégie est la stratégie de ré-optimisation si et seulement si $x_i > 1/2$.
- La méthode de supprimer les objets absents de la solution est une bonne stratégie, si seulement k objets sont présents avec une probabilité :

$$\mathbb{P}(W \leq n-k-1) = 0 \text{ et } \mathbb{P}(W = n-k) > 0$$

alors pour k assez petit (indépendant de n) la stratégie banale est une bonne approche. Mais si k dépend de n, cette méthode est une mauvaise stratégie, en effet pour chaque sous-ensemble S on augmente l'espace perdu dans chaque boîte. Ainsi une redistribution est nécessaire, on va présenter dans les paragraphes suivants des stratégies de redistribution suivant des heuristiques. Pour cela on commence par rappeler les heuristiques existantes pour le PBP.

4.5 Rappel des Heuristiques pour le PBP

Le Bin-packing étant $NP-dur$, les recherches se sont tournées vers l'étude des heuristiques. Un nombre très important d'approximations a été étudié. On va présenter trois classes d'algorithmes (pour plus de détail voir [42]) :

$Next-Fit(NF)$:

- les objets sont placés suivant l'ordre initial de la liste L_n.
- On place x_i (sachant que $x_1, ..., x_{i-1}$ sont déjà affectés) dans la dernière boîte ouverte B_j si possible sinon x_i est placé dansB_{j+1}.

$First-Fit(FF)$:

- les objets sont placés suivant l'ordre initial de la liste L_n.
- On place x_i (sachant que $x_1, ..., x_{i-1}$ sont déjà affectés) dans la première boîte ouverte B_j qui peut le contenir.

$Best-Fit(BF)$:

- les objets sont placés suivant l'ordre initial de la liste L_n.
- l'objet est placé dans la boite qui diminue le plus l'espace perdu (la plus proche s'il y en a deux)

Next-Fit-Decreasing (NFD) (resp First-Fit-Decreasing FFD et Best-Fit-Decreasing BFD) : on ordonne les objets suivant un ordre décroissant puis on applique NF (resp FF et BF).

On remarque que toutes ces règles ont une application particulière. La règle NF (de complexité linéaire) correspond à la situation dans laquelle un ordre de remplissage des boîtes est demandé (B_i est remplit avant B_j si $i < j$) et l'ordre suivant la taille des objets est soit impossible ou très coûteux. Les règles FFD et BFD (plus efficace que NF) sont appliquées à chaque fois que l'ordre suivant la taille des objets est possible et l'ordre de remplissage des boîtes n'est pas important (ou on peut expliciter un ordre des objets de façon que le

placement suivant cet ordre produit la solution de FFD, ou BFD). Le NFD (moins efficace que FFD et BFD) apparaît comme une situation limite de NF où un ordre suivant la taille des objets est possible, mais l'ordre suivant les solutions de FFD et BFD n'est pas possible, cette situation peut se présenter quand la taille des boîtes n'est pas connue et les objets sont ordonnées. Dans ce cas il est plus intéressant de considérer le modèle NFD comme un cas spécial d'un problème plus général où la taille des boîtes varie de façon imprévisible, d'où le packing ne peut être déterminés en avance.

On donne dans le tableau suivant un résumé des performances en pire cas [44] et en moyenne de ces approximations [52] (La moyenne de la solution donnée par l'heuristique dans le cas où les objets sont uniformément et indépendamment répartis dans $[0,1]$).

	complexité	perf pire cas	moyenne de la solution
NF	$0(n)$	2	$\frac{2}{3}n$
FF	$0(nlogn)$	1.691	$1.289\frac{n}{2}$
BF	$0(nlogn)$	$\frac{17}{10}$	$\frac{n}{2}+\theta(\sqrt{n}log^{\frac{3}{4}}n)$
NFD	$0(nlogn)$	$\frac{17}{10}$	$[\frac{\pi^2}{6}-1]n$
BFD	$0(nlogn)$	$\frac{11}{9}$	$\frac{n}{2}+\theta(\sqrt{n})$
FFD	$0(nlogn)$	$\frac{11}{9}$	$\frac{n}{2}+\theta(\sqrt{n})$

Performance des heuristiques pour le Bin-Packing

On remarque que ces algorithmes sont très performants. On va étudier dans les paragraphes suivants des stratégies de redistribution basées sur certaines heuristiques données ci-dessus : on rappelle que la stratégie de redistribution consiste à distribuer les objets présents dans les boîtes suivant une heuristique H donnée.

4.6 Stratégie de Redistribution suivant Next Fit Decreasing

Next Fit Decreasing est l' heuristique suivante.
NFD :

- Sélectionner un ordre décroissant pour les objets.
- Répéter jusqu'à ce que tous les objets soient placés.
  placer l'objet x_i dans la dernière boîte non vide si possible (le niveau de la dernière boîte $+x_i \leq 1$) sinon ouvrir une nouvelle boîte.

On remarque que la méthode de modification de la stratégie de redistribution suivant NFD est linéaire (l'algorithme ci dessus est linéaire) une fois que l'ordre décroissant des objets est préétabli. En plus, NF (NFD) est parmi les rares algorithmes monotones : si on rajoute un objet alors on est ramené à ouvrir au plus une nouvelle boîte ($S' \subset S$ alors $Q^{NF}(S') \leq Q^{NF}(S)$). On dira que la solution est **quasi-stable** si l'absence d'un certain groupe d'objets ne modifie pas la structure de la solution, on montrera que cette propriété est vérifiée par NFD. Suite à ces propriétés et surtout du fait que la solution de NFD est **quasi-stable**, nous avons choisit cette méthode.

4.6.1 Espérance de la Stratégie de Redistribution suivant Next Fit Decreasing

On supposera dans la suite que la liste L_n est donnée suivant un ordre décroissant.

Proposition 38 *Soient n le nombre d'objets et m le nombre de boîtes pour placer la liste L_n des objets suivant NFD. Un objet i a une probabilité de présence p_i, on a alors :*

$$I\!E(\Sigma^{NFD}_{Redist}(L_n)) = \sum_{i=1}^{n} p_i \sum_{l=1}^{m} I\!P(x_i \text{ est au premier niveau de la } l^{\text{ ème }} \text{ boîte})$$

Démonstration :

On remarque qu'on peut construire la stratégie Σ^{NFD}_{Redist} en plaçant les objets un par un, en effet si on note L_i la liste des i premiers objets de L_n, on a alors :

$$\begin{aligned} I\!E(\Sigma^{NFD}_{Redist}(L_i)) &= I\!E(\Sigma^{NFD}_{Redist}(L_i)|x_i \text{ présent})p_i & (4.6)\\ &+ I\!E(\Sigma^{NFD}_{Redist}(L_i)|x_i \text{ absent})(1-p_i) & (4.7)\end{aligned}$$

Or on remarque que

$$I\!E(\Sigma^{NFD}_{Redist}(L_i)|x_i \text{ absent}) = I\!E(\Sigma^{NFD}_{Redist}(L_{i-1}))$$

et

$$I\!E(\Sigma^{NFD}_{Redist}(L_i)|x_i \text{ présent}) = I\!E(\Sigma^{NFD}_{Redist}(L_{i-1})) + I\!P(x_i \text{ est au premier niveau d'une boîte})$$

D'ou

$$I\!E(\Sigma^{NFD}_{Redist}(L_i)) = I\!E(\Sigma^{NFD}_{Redist}(L_{i-1})) + p_i \sum_{l=1}^{m} I\!P(x_i \text{ est au premier niveau de la } l^{\text{ème}} \text{ boîte})$$

∎

On remarque d'après le résultat ci dessus que la contribution d'un objet x_i ne dépend que des objets qui le précèdent (x_j tq $x_j > x_i$). Par l'algorithme NFD on a essentiellement deux catégories de boîtes : des boîtes régulières qui contienne k $k-objet$ et des boîtes frontières qui ne sont pas régulières (contiennent au moins deux objets appartenant à des intervalles différents). Remarquons que l'absence de tous les objets d'une boîte régulière ne change pas la structure de la solution, on dira donc que la solution de NFD est **quasi-stable**. Ceci nous permet d' écrire notre fonctionnelle sous une autre forme :

Proposition 39 *Soit n le nombre d'objets, un objet i a une probabilité de présence p_i, on a alors :*

$$I\!E(\Sigma^{NFD}_{Redist}(L_n)) = \sum_{k\geq 1} I\!E(W_k \, div \, k) + \sum_{k\geq 1} \sum_{s=1}^{k-1} I\!P(A^s_k) \qquad (4.8)$$

où

$$A^s_k = \{\text{les } W_k \bmod k \text{ premiers } k-objet \text{ ne sont pas dans la dernière } s-boîte\}.$$

Démonstration : On note r_k le nombre de $k-boîte$ pour $k \geq 1$, qui est égal au nombre de $k-objets$ mis au premier niveau (bottom) des $k-boîtes$. Rappelons que W_k est le nombre de $k-objet$ présents, par la procédure NFD, on commence par placer les $j_k < k$ premiers $k-objets$ dans la dernière boîte ouverte jusqu'à ce que cette boîte ne peut plus contenir des $k-objets$, on aura après une suite de $k-$boîte régulières $(W_k - j_k \, div \, k)$, enfin on ouvre une boîte pour les $W_k - j_k - k(W_k - j_k \, div \, k) = W_k \, mod \, k - j_k$. Donc le nombre de $k-boîte$ est

$$r_k = \begin{cases} W_k \, div \, k + 1 & si \;\; A_k \\ W_k \, div \, k & sinon \end{cases}$$

où

$$A_k = \{ \text{ les } W_k \, mod \, k \text{ premiers objets ne sont pas dans la dernière non } k-boîte \}$$

On remarque que $A_k = \bigcup_{s<k} A_k^s$ où A_k^s l'événement suivant :

$$A_k^s = \{ \text{ les } W_k \, mod \, k \text{ premiers objets ne sont pas dans la dernière } s-boîte \}$$

Les événements $(A_k^s)_{s<k}$ étant deux à deux disjoints :

$$\mathbb{P}(A_k) = \sum_{s=1}^{k-1} \mathbb{P}(A_k^s)$$

∎

Le premier terme de l'équation (4.8) est une majoration de la moyenne du nombre de boîtes régulières et le deuxième terme est donc une minoration de la moyenne du nombre de boîtes frontières.

4.6.2 Complexité de la Stratégie de Redistribution suivant Next Fit Decreasing

On va étudier maintenant la complexité pour calculer la fonctionnelle de la stratégie de redistribution suivant Next-Fit-Decreasing, on va commencer par donner des cas particuliers.

On remarque que la complexité dépend de la taille des objets et du nombre de $k-objet$ pour chaque k.

Proposition 40 *Si on a que des $k-objet$, pour un k fixé, alors la fonction objective de la stratégie de redistribution suivant $Next-Fit-Decreasing$ se calcule en temps polynomial.*

Démonstration : Supposons qu'on n'a que des $k-objets$, on a d'après la Proposition 39 :

$$\mathbb{E}(\Sigma_{Redist}^{NFD}(L_n)) = \mathbb{E}(W_k div k) + \mathbb{P}(W_k \, mod \, k > 0) \tag{4.9}$$

∎

Si la taille des $k-objets$ est la même pour chaque k, alors on peut calculer la stratégie de redistribution suivant Next Fit Decreasing en un temps polynomial car dans ce cas seulement le nombre de $k-objet$ dans une boîte est important : donc si on connaît ce nombre on peut savoir la structure de la solution.

Proposition 41 *Si la taille des $k-objets$ est $1/k$ pour tout k et x_n est un $l-objet$, l indépendant de n alors la fonction objective de la stratégie de redistribution suivant $Next-Fit-Decreasing$ se calcule en temps polynomial.*

Démonstration :

Nous allons montrer que si la taille des $k-objets$ est égale à $1/k$ pour tout k alors la fonctionnelle s'écrit explicitement.
D'après la Proposition 39, comme x_n est un $l-objet$:

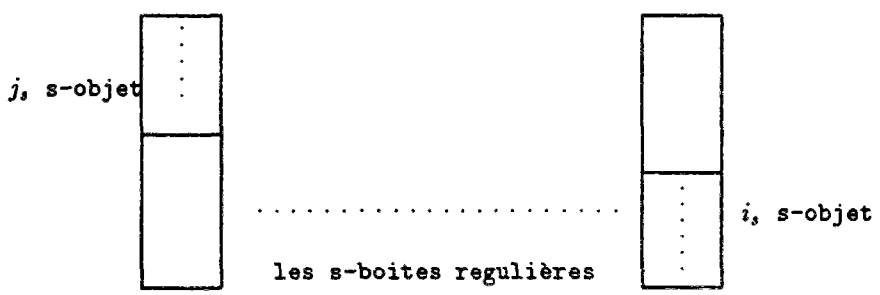
$$\mathbb{E}(\Sigma^{NFD}_{Redist}(L_n)) = \sum_{k=1}^{l} \mathbb{E}(W_k \, div \, k) + \sum_{k=1}^{l}\sum_{s=1}^{k-1} \mathbb{P}(A^s_k)$$

Sachant que le premier terme se calcule en temps polynomial, il suffit de vérifier que le dernier terme se calcule en temps polynomial.

On note n^s_j le nombre de $j-objet$ dans la dernière $s-$boîte. Du fait qu'on ne peut pas mettre plus de s $s-objet$ dans une boîte alors :

$$\mathbb{P}(A^s_k) = \sum_{i_s=1}^{s} \mathbb{P}(A^s_k | n^s_s = i_s)\mathbb{P}(n^s_s = i_s) \tag{4.10}$$

Sachant que n^s_s dépend que du nombre d'objet dans la dernière boîte frontière avant les $s-$boîte régulières (le nombre de boîte régulière n'est pas important) :



$$\mathbb{P}(n^s_s = i_s) = \sum_{s'=1}^{s-1}\sum_{j_s=1}^{s-1} \mathbb{P}(n^s_s = i_s | n^{s'}_s = j_s)\mathbb{P}(n^{s'}_s = j_s) \tag{4.11}$$

Comme $W_s - n^s_s - n^{s'}_s$ est le nombre de $s-objet$ dans les boîtes régulières donc il doit être divisible par s :$W_s - n^s_s - n^{s'}_s \; mod \; s = 0$

$$\mathbb{P}(n^s_s = i_s | n^{s'}_s = j_s) = \mathbb{P}(W_s \; mod \; s = (i_s + j_s) \; mod \; s) \tag{4.12}$$

D'après les équations (4.12),(4.11) si $\mathbb{P}(n_s^{s'} = j_s)$ se calcule en un temps polynomial alors il en est de même pour $\mathbb{P}(n_s^s = i_s)$. On remarque que le calcul de $\mathbb{P}(n_s^{s'} = j_s)$ se fait de la même façon que celui de $\mathbb{P}(A_k^s | n_s^s = i_s)$ que nous allons expliciter.

Pour calculer $\mathbb{P}(A_k^s | n_s^s = i_s)$, on conditionne par rapport au nombre de $j - objet$ présent dans la dernière $s -$ boîte pour $s < j < k$. Pour que la dernière $s -$ boîte contienne des $k - objet$, il faut que tous les $j - objets$ présent pour $s < j < k$ se placent dans la dernière $s -$ boîte. On obtient donc $k - 1 - s$ sommes :

$$\mathbb{P}(A_k^s | n_s^s = i_s) = \sum_{(i_{s+1},...,i_{k-1}) \in L_{s,k}} \mathbb{P}(A_k^s | n_s^s = i_s \text{et } \{W_j = W_j \ mod \ j = n_j^s = i_j\}) \prod_{j=s+1}^{k-1} \mathbb{P}(W_j = W_j \ mod \ j = n_j^s = i_j)$$

où

$$L_{s,k} = \{(i_{s+1}, ..., i_{k-1}) \text{ tq } 0 \leq i_j \leq j^*\}$$

où j^* est le maximum de $j - objet$ qu'on peut mettre dans $1 - \sum_{e=s}^{j-1} \frac{i_e}{e}$.

Il est clair que $\mathbb{P}(W_j = W_j \ mod \ j = n_j^s = i_j)$ se calcule en temps polynomial. Donc il reste à montrer que $\mathbb{P}(A_k^s | n_s^s = i_s \text{et } \{W_j = W_j \ mod \ j = n_j^s = i_j\})$ se calcule en temps polynomial. Comme les $j - objets$ sont tous de taille $1/j$ pour tout j, alors le niveau de la dernière $s -$ boîte est égale à $\sum_{j=s}^{k-1} \frac{i_j}{j}$. Soit k^* le maximum de $k - objet$ qu'on peut mettre dans $1 - \sum_{j=s}^{k-1} \frac{i_j}{j}$:

$$\mathbb{P}(A_k^s | \bigcap_{j=s+1}^{k-1} \{W_j = W_j \ mod \ j = n_j^s = i_j\}) = \mathbb{P}(W_k \ mod \ k = k^*)$$

∎

Si les objets sont de tailles différentes, alors cette démonstration n'est plus valable car on ne peut plus conditionner par rapport au nombre d'objet du fait que pour un même nombre de $k - objet$ on n'a plus le même espace occupé.

Théorème 18 *La fonction objective de la stratégie de redistribution suivant $Next - Fit - Decreasing$ ne se calcule pas en temps polynomial.*

Démonstration : On va montrer que la fonctionnelle $\mathbb{E}(\Sigma_{Redist}^{NFD}(L_n))$ ne se calcule pas en un temps polynomial.

On note h^i le niveau de la dernière boîte quand les objets présents de $L_i = (x_1, ..., x_i)$ sont placés, h^i est une variable aléatoire définie par

$$h^i = \begin{cases} h^{i-1} + x_i & si \quad h^{i-1} + x_i \leq 1 \quad x_i \text{ présent} \\ h^{i-1} & si \quad x_i \text{ absent} \\ x_i & si \quad h^{i-1} + x_i > 1 \quad \text{et } x_i \text{ présent} \end{cases}$$

On ouvre une nouvelle boîte pour placer x_i si et seulement si x_i est présent et $h^{i-1} + x_i > 1$, alors d'après la Proposition 38, notre fonctionnelle peut s'écrire de la façon suivante :

$$\mathbb{E}(\Sigma^{NFD}_{Redist}(L_n)) = \sum_{i=1}^{n} p_i \mathbb{P}(h^{i-1} + x_i > 1) \tag{4.13}$$

Nous allons montrer que $\mathbb{P}(h^{i-1} + x_i > 1)$ ne se calcule pas en temps polynomial,

$$\begin{aligned} \mathbb{P}(h^{i-1} > 1 - x_i) &= p_i \mathbb{P}(h^{i-2} + x_{i-1} > 1 - x_i)(1 - \mathbb{P}(h^{i-2} + x_{i-1} > 1)) \\ &+ (1 - p_i)\mathbb{P}(h^{i-2} > 1 - x_i) \\ &+ p_i \mathbb{P}(h^{i-2} > 1 - x_{i-1}) 1_{\{x_{i-1} > 1 - x_i\}} \end{aligned} \tag{4.14}$$

Si on note $C_i(x)$ la complexité de $\mathbb{P}(h^{i-1} > x)$ on a alors d'après (4.14) :

$$C_i(1 - x_i) = C_{i-1}(1 - x_i - x_{i-1}) + C_{i-1}(1 - x_{i-1}) + C_{i-1}(1 - x_i)$$

Donc par récurrence on a le résultat. ∎

4.6.3 Bornes de la Stratégie de Redistribution suivant Next Fit Decreasing

On va donner une inégalité qui nous permettra de mieux comprendre le comportement de la stratégie de redistribution suivant NFD.

Théorème 19 *Soit n le nombre d'objets, supposons que tous les objets ont une même probabilité de présence p. Soit n_k le nombre de $k-objet$ dans la liste L_n on a alors :*

$$| \mathbb{E}(\Sigma^{NFD}_{Redist}(L_n)) - \sum_{k \geq 1} p \frac{n_k}{k} | \leq 1 + lognp \tag{4.15}$$

Démonstration : On rappelle l'inégalité déterministe de RHEE [57], on note $n_k(S)$ le nombre de $k-objet$ dans la liste $S \subseteq L_n$:

$$|Q^{NFD}_n(S) - \sum_{k \geq 1} \frac{n_k(S)}{k}| \leq 1 + log|S| \quad \forall \quad S \tag{4.16}$$

L'inégalité (4.15) peut être déduite directement (4.16) et de l'inégalité de Jensen. ∎

Remarques :

- On remarque que $\sum_{k \geq 1} p \frac{n_k}{k}$ correspond à une majoration de la moyenne de la taille des objets et $1 + lognp$ est une borne des boîtes frontières.
- Quand p est grand (proche de 1) on retrouve le même comportement que la solution NFD quand tous les objets sont présents. Dans le cas de p très petit la stratégie ne diffère pas beaucoup de la moyenne de la taille des objets.

4.6.4 Relation entre la Stratégie de Redistribution suivant NFD et la Solution Optimale

On va donner des bornes de la stratégie de Redistribution suivant NFD. On commence par rappeler l'étude du comportement en pire cas de NFD [2]. Quand la taille du plus grand objet est un paramètre, on a besoin des deux suites suivantes : $t_i(q)$ est la suite définie par

$$t_1(q) = q+1,\ t_2(q) = q+2$$
$$t_{i+1}(q) = t_i(q)[t_i(q)-1]+1 \quad \forall i \geq 2$$

Et la suite $\{a_i(q)\}$ définie par $a_i(q) = t_i(q) - 1$, $\forall i \geq 1$, on définit alors

$$\gamma_q = \sum_{i=1}^{\infty} \frac{1}{a_i(q)} = \sum_{i=1}^{\infty} \frac{1}{(t_i(q)-1)}$$

Théorème 20 (Coffman 1981) *Soient $L_n = (x_1, ..., x_n)$ une liste d'objet tel que $x_1 \geq ... \geq x_n$, et q le plus petit entier tel que $x_1 \in (\frac{1}{q+1}, \frac{1}{q}]$. Alors*

$$NFD(L_n) \leq \gamma_q^* OPT(L_n) + 3 \tag{4.17}$$

où $\gamma_q^* = \frac{q-1}{q} + \gamma_q$. La borne (4.17) est atteinte (la plus petite possible).

Démonstration : On va donner l'idée de la démonstration :

on définit une fonction de pondération sur la taille des objets qu'on note $F_q(x)$, on note le poids cumulatif des objets $F_q(L_n) = \sum_{i=1}^{n} F_q(x_i)$, on démontre que le poids cumulatif vérifie les deux propriétés suivantes (ceci est le schéma pour toutes les analyses en pire cas pour le bin packing, pour chaque heuristique on a une fonction de pondération particulière) :

$\mathcal{G}_1)$ $NFD(L_n) - 3 \leq F_q(L_n)$

$\mathcal{G}_2)$ $F_q(L_n) \leq \gamma_q^* OPT(L_n)$

La fonction pondération pour NFD est définie comme suit : si x est un $k-objet$, $k \geq q$,

$$F_q(x) = \begin{cases} \frac{1}{k} & \text{si } k = t_i(q) - 1 \text{ pour un } i \geq 1 \\ \frac{k+1}{k} x & \text{sinon} \end{cases}$$

∎

Remarques :

- On remarque que $\sum_{i=1}^{\infty} \frac{1}{t_i(q)} = \frac{2}{q+1}$ et $\sum_{i=k}^{\infty} \frac{1}{t_i(q)} = \frac{1}{t_k(q)-1}$.
- γ_q^* est décroissante : les premières valeurs de γ_q^* sont $\gamma_1^* = 1,691...$, $\gamma_2^* = 1,423...$, $\gamma_3^* = 1,302...$, $\gamma_4^* = 1,233....$ On remarque que γ_q^* approche de 1 avec une croissance presque égale à $\frac{q+2}{q+1}$ en effet on a $\gamma_q^* = \frac{q+2}{q+1} + \sum_{i=3}^{\infty} \frac{1}{a_i(q)}$

- La borne (4.17) est atteinte (on donne un exemple où $\lim_{n\longrightarrow\infty} \frac{NFD(L_n)}{OPT(L_n)} \geq \gamma_q^*$) . Mais d'après la convergence rapide de la série $\sum_{i=1}^{\infty} \frac{1}{t_i(q)}$ on a besoin seulement de petites valeurs de n pour avoir une approximation de γ_q^* (si $q = 1$ alors pour $n \geq 5$ on a la convergence).

En utilisant le Théorème ci dessus on va donner le comportement de notre stratégie de redistribution suivant NFD par rapport à la solution optimale ($OPT(L_n)$) :

Proposition 42 *Soient $L_n = (x_1, ..., x_n)$ une liste d'objets tel que $x_1 \geq ... \geq x_n$, et q le plus petit entier tel que $x_1 \in (\frac{1}{q+1}, \frac{1}{q}]$ et chaque objet est présent avec une probabilité p . Alors*

$$I\!E(\Sigma_{Redist}^{NFD}(L_n)) \leq p\gamma_q^* OPT(L_n) + 3 \tag{4.18}$$

où $\gamma_q^* = \frac{q-1}{q} + \gamma_q$,

Démonstration : On va reprendre la même fonction de pondération proposée par Coffman pour l'étude de NFD. On a d'après $\mathcal{G}_1$:

$$I\!E(\Sigma_{Redist}^{NFD}(L_n)) - 3 \leq \sum_S I\!P(S) F_q(S)$$

On remplace $F_q(S)$ par sa valeur :

$$I\!E(\Sigma_{Redist}^{NFD}(L_n)) - 3 \leq \sum_S I\!P(S) \sum_{x_i \in S} F_q(x_i) \tag{4.19}$$

Si on note S_{x_i} les sous ensembles d'objets contenant x_i alors :

$$\sum_S I\!P(S) \sum_{x_i \in S} F_q(x_i) = \sum_{i=1}^{n} F_q(x_i) \sum_{S_{x_i}} I\!P(S_{x_i}) = pF_q(L_n) \tag{4.20}$$

$$I\!E(\Sigma_{Redist}^{NFD}(L_n)) - 3 \leq pF_q(L_n) \tag{4.21}$$

le résultat découle d'après (4.21) et la propriété $\mathcal{G}_2$. ∎

4.6.5 Relation entre la Stratégie de Redistribution suivant NFD et la Stratégie de Ré-optimisation

Nous déduisons d'après le Théorème 20 une borne de la stratégie de redistribution. On note $x_1(S)$ l'objet de plus grande taille dans S.

Proposition 43 *Soient $L_n = (x_1, ..., x_n)$ une liste d'objet tel que $x_1 \geq ... \geq x_n$, et chaque objet est présent avec une probabilité p. Soit $q(S)$ la variable aléatoire qui est égale au plus petit entier tel que $x_1(S) \in (\frac{1}{q(S)+1}, \frac{1}{q(S)}]$. Alors :*

$$I\!E(\Sigma_{Redist}^{NFD}(L_n)) \leq \sum_S I\!P(S)\gamma_{q(S)}^* NFD(S) + 3 \tag{4.22}$$

où $\gamma_{q(S)}^* = \frac{q(S)-1}{q(S)} + \gamma_{q(S)}$,

Le fait que la suite γ_q^* est décroissante nous déduisons de (4.22) une borne de la stratégie de redistribution par rapport à la stratégie de ré-optimisation :

Proposition 44 *Soient $L_n = (x_1, ..., x_n)$ une liste d'objet tel que $x_1 \geq ... \geq x_n$, et q le plus petit entier tel que $x_1 \in (\frac{1}{q+1}, \frac{1}{q}]$ et chaque objet est présent avec une probabilité p. Alors :*

$$I\!E(\Sigma^{NFD}_{Redist}(L_n)) \leq \gamma_q^* \; I\!E(\Sigma^{OPT}_{Redist}(L_n)) + 3 \tag{4.23}$$

ou $\gamma_q^* = \frac{q-1}{q} + \gamma_q$,

4.6.6 Stratégie de Redistribution de NFD Aléatoires

On est intéressé dans ce paragraphe par une analyse en moyenne du comportement de la stratégie de Redistribution suivant NFD. Pour ceci on va supposer que la taille des objets est une variable aléatoire. Afin d'exposer les résultats suivants, on introduit les notations suivantes

- Soit $x = \{x_1, x_2, ...\}$ une suite de point de $[0, 1]$ et $x^n = \{x_1, ..., x_n\}$. Si la position des points est aléatoire la suite est représentée par $X = \{X_1, X_2, ...\}$.
- On pourra introduire l'espace de probabilités $(\Omega, \mathcal{A}, \mathbf{P})$ où $\Omega := [0, 1]$ et $\mathcal{A}$ est sa tribu Borélienne sur lequel les X_i sont des v.a et on définira les suites $X^{(n)}$ sur l'espace induit $(\Omega^{I\!N}, \mathcal{A}^{I\!N}, \mathbf{P}^{I\!N})$.
- Pour éviter toute confusion on notera $\mathbf{Eu}(.)$ l'espérance relative à la mesure de probabilités $\mathbf{P}$.

On remarque que $I\!E(\Sigma^{NFD}_{Redist}(X_1, ..., X_n))$ est une variable aléatoire par rapport à la taille des objets.

Théorème 21 *Supposons que $X_1, ..., X_n$ sont indépendantes, identiquement distribués suivant une loi μ et que les objets sont présents avec une probabilité p. Pour $k \geq 1$ et $\alpha_k = \mu((\frac{1}{k+1}, \frac{1}{k}])$. Soit $\theta = \sum_{k\geq 1} \frac{\alpha_k}{k}$. Alors*

i) $\lim_{n\to\infty} n^{-1} \; I\!E(\Sigma^{NFD}_{Redist}(X_1, ..., X_n)) = p\theta. \quad p.s$

ii)

$$\frac{I\!E(\Sigma^{NFD}_{Redist}(X_1, ..., X_n)) - np\theta}{p\sqrt{n}} \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sum_{k\geq 1} \frac{\alpha_k}{k^2 - \theta^2})$$

iii) Pour tout $t \geq (1 + \log n)/n$

$$I\!P(\; I\!E(\Sigma^{NFD}_{Redist}(X_1, ..., X_n)) - np\theta > tn) \leq 2exp(-2n(t - (1 + \log(n))/n)^2) \tag{4.24}$$

Démonstration : $\mathbb{E}(\Sigma^{NFD}_{Redist}(X_1,...,X_n))$ est asymptotiquement équivalente à la somme de variable aléatoire i.i.d en effet on a d'après le Théorème (19) :

$$| \mathbb{E}(\Sigma^{NFD}_{Redist}(X_1,...,X_n) - \sum_{k\geq 1} p\frac{n_k}{k}| \leq 1 + \log(np)$$

On considère la fonction f (proposée par RHEE [57]) sur $[0,1]$ définie par :

$$f(x) = \begin{cases} \frac{1}{k} & \text{si x est un k-objet et x présent.} \\ 0 & \text{si } x = 0 \end{cases}$$

Comme $n_k = \sum_{i=1}^{n} 1_{\{x_i \text{est un } k-objet\}}$ d'où :

$$\sum_{k\geq 1} p\frac{n_k}{k} = p\sum_{k\geq 1}\frac{1}{k}\sum_{i=1}^{n} 1_{\{x_i k-objet\}} = p\sum_{i=1}^{n} f(x_i)$$

Il suit donc que

$$| \mathbb{E}(\Sigma^{NFD}_{Redist}(X_1,...,X_n)) - p\sum_{i=1}^{n} f(X_i)| \leq 1 + \log(np) \tag{4.25}$$

Les v.a $f(X_i)$ sont i.i.d donc on a ramené le problème à l'étude de somme de va i.i.d. On pose $S_n = p\sum_{i=1}^{n} f(X_i)$, d'après la loi des grand nombre

$$\lim_{n\longrightarrow\infty} \frac{S_n}{n} = p\mathbf{Eu}(f(X_i)) \quad p.s$$

Comme X_i est distribué suivant μ on a :

$$\mathbf{Eu}(f(X_i)) = \sum_{k\geq 1}\frac{1}{k}\mu((\frac{1}{k+1},\frac{1}{k}]) = \theta$$

D'après les deux équations ci dessus et (4.25) on a $\lim_{n\to\infty} n^{-1}\ \mathbb{E}(\Sigma^{NFD}_{Redist}(X_1,...,X_n)) = p\theta.$ $p.s.$ La loi de notre fonctionnelle peut être déduite d'après le théorème limite central. En effet,

$$\frac{S_n - np\theta}{\sigma\sqrt{n}} \xrightarrow{\mathcal{L}} \mathcal{N}(0;1),$$

où $\mathcal{N}(0;1)$ est la loi normale centrée réduite, et σ^2 est la variance de $pf(X_i)$:

$$\sigma^2 = \sum_{k\geq 1}(\frac{p}{k})^2\mu((\frac{1}{k+1},\frac{1}{k}]) - (\sum_{k\geq 1}\frac{p}{k}\mu((\frac{1}{k+1},\frac{1}{k}]))^2 \tag{4.26}$$

$$= p^2[\sum_{k\geq 1}(\frac{1}{k})^2\mu((\frac{1}{k+1},\frac{1}{k}]) - (\sum_{k\geq 1}\frac{1}{k}\mu((\frac{1}{k+1},\frac{1}{k}]))^2] \tag{4.27}$$

D'après [57] on a :

$$\sigma^2 = p^2\sum_{k\geq 1}\frac{\alpha_k}{k^2-\theta^2}$$

Ainsi d'après l'équation (4.25) et la convergence en loi de $\frac{S_n - np\theta}{\sigma\sqrt{n}}$, on déduit la convergence ii)

$$\frac{\mathbb{E}(\Sigma^{NFD}_{Redist}(X_1, ..., X_n)) - np\theta}{p\sqrt{n}} \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sum_{k\geq 1} \frac{\alpha_k}{k^2 - \theta^2})$$

Enfin on s'intéresse à une estimation de la probabilité de la déviation de notre fonctionnelle de sa moyenne. Comme nos variables aléatoires $f(X_i)$ sont bornées iii) est une conséquence de l'inégalité de Hoeffding qu'on rappelle :

$$\mathbb{P}(S_n - \mathbb{E}(S_n) > tn) \leq exp(-2n(t - \frac{(1 + logn)}{n})^2) \quad \text{pour } tn > 1 + logn$$

pour plus de détail sur ces résultats voir [21]. ∎

Remarques :

- Dans le cas particulier où les objets sont uniformément réparties sur $[0, 1]$ alors $\mathbb{E}(f(X_i)) = \frac{\pi^2}{6} - 1$ on a donc :

$$\lim_{n\longrightarrow\infty} n^{-1}\ \mathbb{E}(\Sigma^{NFD}_{Redist}(X_1, ..., X_n)) = p(\frac{\pi^2}{6} - 1) \quad p.s$$

- La solution obtenue par NFD à travers la moyenne des objets ($\mathbb{E}(W) = np$) est une assez bonne approximation de notre fonctionnelle $\mathbb{E}(\Sigma^{NFD}_{Redist}(X_1, ..., X_n))$ si $X_1, ..., X_n$ sont i.i.d suivant une loi μ.

4.7 Stratégie de Redistribution suivant Matching

Matching est un algorithme introduit ([23], [48], [51]) pour l'étude asymptotique de la solution optimale quand les éléments de la liste L_n sont uniformément réparties sur $[0, 1]$. Matching est l' heuristique suivante.

Matching :

- Répéter jusqu'à ce que tous les objets soient placés.
  - L_i la liste des objets à placer
  - Sélectionner y le plus grand objet de L_i.
    - Sélectionner x le plus grand objet de $L_i \backslash \{y\}$ tel que $x + y \leq 1$, placer (x,y) dans une même boîtes.
    - Si un tel x n'existe pas placer y seul dans une boîtes

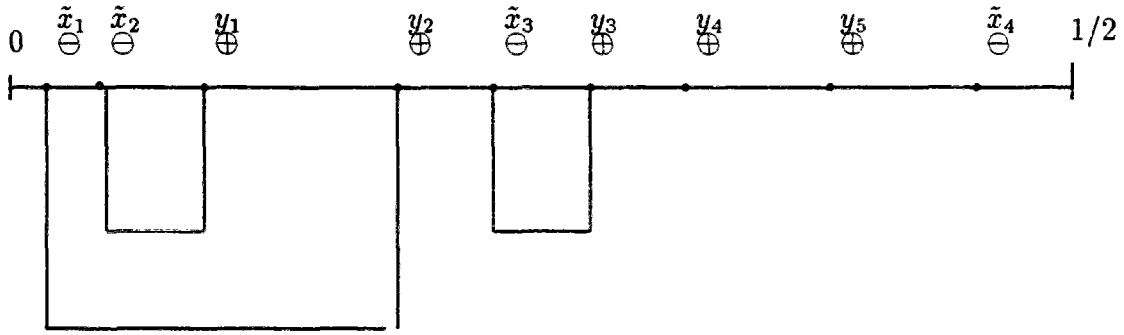
L'étude la plus élégante de cet algorithme est celle de Karp [52], il réduit le problème à la recherche de l'excès des grands objets sur les petits : pour cela on représente chaque petit objet x par $\ominus$ au point x et pour chaque grand objet x un $\oplus$ au point $1 - x$. On représente la liste d'objets sur $[0, \frac{1}{2}]$ par la méthode ci dessus. Les objets de taille $\leq 1/2$ (resp $> 1/2$) seront dits petits (resp grands) objets ; de même pour les boîtes singletons. On remarque que le nombre de boîtes utilisés est égal au nombre d'objets plus le nombre de boîtes singleton

divisé par deux. Il est clair qu'un grand objet ne peut être affecté qu'avec des petits objets qui sont à sa gauche (puisque $1-x$ correspond à l'espace perdu quand on place l'objet x). On note $Exc(x)$ l'excès des grands objets sur les petits au point x c'est à dire la différence du nombre des $\oplus$ sur les $\ominus$ qui sont à gauche de x. On remarque que le nombre de grandes boîtes singleton est le maximum de l'excès :$\max_x Exc(x)$.

Cette méthode nous permet d'avoir un ordre d'affectation des objets intéressant. Cet ordre rend la solution du *Matching* **quasi-stable**, en effet quand un grand objet pour une liste donnée et le petit objet associé sont absents le nombre de boîte singleton reste le même. La quasi-stabilité nous permet d'écrire explicitement notre fonctionnelle. En plus cet algorithme est asymptotiquement équivalent à la solution optimale quand les éléments de la liste L_n sont uniformément réparties sur $[0,1]$ (voir [52]), ceci nous permettra de donner le comportement asymptotique de la stratégie de ré-optimisation.

4.7.1 Espérance de la Stratégie de Redistribution suivant Matching

On note b_1 (resp $b_1(S)$)le nombre de boîte singleton (pour la réalisation S), $Match(S)$ le nombre de boîtes utilisées par l'heuristique *Matching* pour la réalisation S, $\tilde{x}_i$ le $i^{\text{ème}}$ petit objet et y_i le $i^{\text{ème}}$ grand objet dans la représentation des objets sur $[0, 1/2]$. On rappelle que n_1 est le nombre de grand objet(nombre de $1-objet$).



Exemple pour Matching

Théorème 22 *Soit n le nombre d'objets. Un objet i a une probabilité de présence p, on a alors :*

$$\begin{aligned} I\!E(\Sigma^{Match}_{Redist}(L_n)) &= \frac{np}{2} \\ &+ \frac{p}{2}\sum_{j=1}^{n_1} I\!P(y_j \text{ seul dans une boîte}) \\ &+ \frac{p}{2}\sum_{i=1}^{n-n_1} I\!P(\tilde{x}_i \text{ est dans une petite boîte singleton}) \end{aligned} \tag{4.28}$$

Démonstration : Sachant que les boîtes du packing produit par *Matching* contiennent soit un objet ou deux objets, on note $b_1(S)$ le nombre de boîte singleton pour l'exemplaire S, alors on a :

$$Match(S) = \frac{|S| + b_1(S)}{2} \quad \forall\ S \tag{4.29}$$

$|S|$ suit une loi binomiale $\mathcal{B}(n,p)$:

$$\mathbb{E}(\Sigma^{Match}_{Redist}(L_n)) = \frac{np}{2} + \frac{\mathbb{E}(b_1)}{2} \tag{4.30}$$

Le problème se ramène à évaluer la moyenne du nombre de boîtes singletons. Alors il est clair qu'on a au plus une seule petite boîte singleton. Si on note $\tilde{b}_1$ le nombre de grande boîte singleton on a :

$$\mathbb{E}(b_1) = \mathbb{E}(\tilde{b}_1) + \mathbb{P}(\exists \text{ petite boîte singleton}) \tag{4.31}$$

Or

$$\{\exists \text{petite boîte singleton}\} = \bigcap_{i=1}^{n-n_1} \{\tilde{x}_i \text{ est dans une petite boîte singleton}\}$$

ces événements étant deux à deux disjoints :

$$\mathbb{P}(\exists \text{petite boîte singleton}) = \sum_{i=1}^{n-n_1} p\mathbb{P}(\tilde{x}_i \text{est dans une petite boîte singleton})$$

$$\mathbb{E}(\tilde{b}_1) = \sum_{i=1}^{n_1} p\mathbb{P}(y_i \text{ seul dans une boîte}) \tag{4.32}$$

∎

Nous pouvons calculer explicitement notre fonctionnelle (4.28). On introduit les notations suivantes :

- m_j le nombre de petits objets compris entre y_{j-1} et y_j pour $j \geq 1$ et m_1 le nombre de petits objets compris entre 0 et y_1.
- On note τ_j $(v.a)$ le $j^{\text{ème}}$ grand objet présent, $\tau_0 = 0$:

$$\tau_j = \min_k\{k > \tau_{j-1} / y_k \text{présent }\}$$

Lemme 6

$$I\!P(y_{\tau_j} \textit{seul dans une boîte}) = \sum_{(i_0,\ldots,i_{j-1}) \in I_j} \prod_{k=0}^{j-1} I\!P(A_{ji_k}) \tag{4.33}$$

où

$$I_j = \{(i_0,\ldots,i_{j-1}) | i_0 = 0 \textit{ et } 0 \leq i_k \leq k - \sum_{e=1}^{k-1} i_e \textit{ pour } 1 \leq k \leq j-1\}$$

et

$$A_{ji_k} = \{i_k \textit{présent parmi } m_{\tau_{j-k-1}+1} + \ldots + m_{\tau_{j-k}}\} \quad \textit{pour} \quad 0 \leq k \leq j-1$$

Démonstration :

y_{τ_j} est dans une boîte singleton s'il est présent, si tous les petits objets présents avant y_{τ_j} sont affectées et tous les petits objets entre $y_{\tau_{j-1}}$ et y_{τ_j} sont absents. On note i_l le nombre de petits objets présent parmi les objets compris entre $y_{\tau_{j-l-1}}$ et $y_{\tau_{j-l}}$, on pose

$$A_{j,i_0,\ldots,i_{j-1}} = \{y_{\tau_j}\text{seul }\} \bigcap_{l=0}^{j-1} \{i_l\text{présent entre } y_{\tau_{j-l-1}}\text{et } y_{\tau_{j-l}}\}$$

$$\mathbb{P}(y_{\tau_j}\text{seul dans une boîte}) = \mathbb{P}(\bigcup A_{j,i_0,\ldots,i_{j-1}})$$

Les événements $A_{j,i_0,\ldots,i_{j-1}}$ étant deux à deux disjoints

$$\mathbb{P}(y_{\tau_j}\text{seul dans une boîte}) = \sum_{i_0,\ldots,i_{j-1}} \mathbb{P}(A_{j,i_0,\ldots,i_{j-1}}) \tag{4.34}$$

On remarque que y_{τ_j} est seul dans une boîte ssi i_k est au maximum égal au nombre de grands objets non affectés entre $y_{\tau_{j-k-1}}$ et $y_{\tau_{j-1}}$, pour $k \geq 1$, ceci se traduit par $(i_0, \ldots, i_{j-1}) \in I_j$ où :

$$I_j = \{(i_0, \ldots, i_{j-1}) | i_0 = 0 \text{ et } 0 \leq i_k \leq k - \sum_{l=0}^{k-1} i_l \text{ pour } 1 \leq k \leq j-1\} \tag{4.35}$$

On note A_{ji_k}

$$\begin{aligned} A_{ji_k} &= \{i_k\text{petits objets présents entre } y_{\tau_{j-k-1}}\text{et } y_{\tau_{j-k}}\} \\ &= \{i_k\text{petits objets présents parmi } m_{\tau_{j-k-1}+1} + \ldots. + m_{\tau_{j-k}}\} \end{aligned}$$

Sous la condition (4.35), y_{τ_j} est seul dans une boîte donc :

$$A_{j,i_0,\ldots,i_{j-1}} = \bigcap_{k=0}^{j-1} A_{ji_k}$$

Or ces événements étant mutuellement indépendants :

$$\mathbb{P}(A_{j,i_0,\ldots,i_{j-1}}) = \prod_{k=0}^{j-1} \mathbb{P}(A_{ji_k})$$

∎

Remarque :

- Le calcul de $\mathbb{E}(\tilde{b}_1)$ se déduit du Lemme précédent.

Lemme 7

$$I\!E(\tilde{b_1}) = \sum_{l=0}^{n_1} \sum_{(t_1,...,t_l)\in B_t^l} I\!E(\tilde{b_1}|y_{t_1},....,y_{t_l}présents\ )p^l(1-p)^{n_1-l} \tag{4.36}$$

$$B_t^l = \{(t_1,...,t_l)|k \leq t_k \leq t_{k+1} pour\ 1 \leq k \leq l-1 et\ l \leq t_l \leq n_1\}$$

et

$$I\!E(\tilde{b_1}|y_{t_1},...,y_{t_l}présents\ ) = \sum_{j=1}^{l} I\!P(y_{t_j} seul\ dans\ une\ boîte|y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l})$$

Démonstration : On conditionne par $(y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l})$:

$$\mathbb{E}(\tilde{b_1}) = \sum_{l=0}^{n_1} \sum_{t_1,...,t_l} \mathbb{E}(\tilde{b_1}|y_{t_1},...,y_{t_l}\text{présents }) p^l(1-p)^{n_1-l}$$

sous la condition $(t_1,...,t_l) \in B_t^l$ où

$$B_t^l = \{(t_1,...,t_l)|k \leq t_k \leq t_{k+1}\text{pour } 1 \leq k \leq l-1\text{et } l \leq t_l \leq n_1\}$$

On remarque que

$$\begin{aligned} \mathbb{E}(\tilde{b_1}|y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l}) &= \mathbb{E}(\sum_{j=1}^{l} 1_{\{y_{t_j}\text{seul dans une boîte}\}|y_{\tau_1}=y_{t_1},....,y_{\tau_l}=y_{t_l}}) \quad (4.37) \\ &= \sum_{j=1}^{l} \mathbb{P}(y_{t_j}\text{seul dans une boîte}|y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l}) \end{aligned}$$

∎

- On remarque que $\mathbb{P}(y_{t_j}\text{seul dans une boîte}|y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l})$ est donnée explicitement par le lemme 6 : i_k suit une loi binomiale $\mathcal{B}(m_{t_{j-k-1}+1} + ... + m_{t_{j-k}}, p)$.
- On note b_1^p le nombre de petites boîtes singletons (v.a qui vaut 1 ou 0).

Lemme 8

$$I\!E(b_1^p) = \sum_{l=0}^{n_1} \sum_{(t_1,...,t_l)\in B_t^l} I\!E(b_1^p|y_{\tau_1} = y_{t_1},...,y_{\tau_l} = y_{t_l})p^l(1-p)^{n_1-l} \tag{4.38}$$

$$B_t^l = \{(t_1,...,t_l)|k \leq t_k \leq t_{k+1} pour\ 1 \leq k \leq l-1 et\ l \leq t_l \leq n_1\}$$

$$I\!E(b_1^p|y_{t_1},...,y_{t_l}présents\ ) = \sum_{j=1}^{l} \sum_{i=1}^{m_{t_{j-1},t_j}} I\!E(1_{\{x_{t_j^i} seul\ dans\ une\ boîte|y_{\tau_1}=y_{t_1},....,y_{\tau_l}=y_{t_l}\}})$$

où $m_{t_{j-1},t_j} = \sum_{e=1}^{t_j-t_{j-1}} m_{t_{j-1}+e}$

Démonstration :

On conditionne par $(y_{\tau_1} = y_{t_1}, ..., y_{\tau_l} = y_{t_l})$:

$$\mathbb{E}(b_1^p) = \sum_{l=0}^{n_1} \sum_{t_1,...,t_l} \mathbb{E}(b_1^p | y_{t_1}, ..., y_{t_l} \text{présents }) p^l (1-p)^{n_1 - l}$$

On note le nombre de petit objet entre $y_{t_{j-1}}$ et y_{t_j} m_{t_{j-1},t_j} : $m_{t_{j-1},t_j} = \sum_{e=1}^{t_j - t_{j-1}} m_{t_{j-1}+e}$

$$\mathbb{E}(b_1^p | y_{t_1}, ..., y_{t_l} \text{présents }) = \sum_{j=1}^{l} \sum_{i=1}^{m_{t_{j-1},t_j}} \mathbb{E}(1_{\{x_{t_j^i} \text{seul dans une boîte} | y_{\tau_1} = y_{t_1}, ..., y_{\tau_l} = y_{t_l}\}})$$

∎

On peut calculer $\mathbb{E}(1_{\{x_{t_j^i} \text{seul dans une boîte} | y_{\tau_1} = y_{t_1}, ..., y_{\tau_l} = y_{t_l}\}})$ explicitement (pour plus de détail voir appendice A).

Complexité de la Stratégie de Redistribution suivant Matching.

Théorème 23 *La fonction objective de la stratégie de redistribution suivant Matching ne se calcule pas en temps polynomial.*

Démonstration :

Si on suppose que le calcul de $\prod_{k=0}^{j-1} \mathbb{P}(A_{ji_k})$ se fait en un temps constant, alors d'après le Lemme 6 le calcul de $\mathbb{P}(y_{\tau_j} \text{seul dans une boîte})$ se fait en un temps de l'ordre du nombre de $j-uplet$ dans I_j $(2^{j-1} \leq |I_j| \leq j-1!)$, or $|I_j|$ est de l'ordre $0(\frac{(2j+1)!}{(j+2)(j-1)!})$ (on rappelle que $\sum_{j=1}^{n} j(j+1)...(j+k) = \frac{(n+k+1)!}{(k+2)(n-1)!}$). Ainsi soit il existe un y_j où $j = 0(n)$ alors l'équation (4.33) se calcule au moins en $0(n!)$, sinon il existerai un $\tilde{x}_i$ où $i = 0(n)$, et le calcul de l'équation (4.38) est de l'ordre de $0(n!)$. ∎

4.7.2 Stratégie de Redistribution de Matching Aléatoire

On se place maintenant dans le cas où la taille des objets est une variable aléatoire. On s'intéresse en particulier au comportement asymptotique de la fonctionnelle $\mathbf{Eu}(\ \mathbb{E}(\Sigma_{Redist}^{Match}(L_n)))$

Théorème 24 *Supposons que $X_1, ..., X_n$ sont indépendantes, identiquement distribués suivant une loi uniforme sur $[0,1]$ et que les objets sont présents avec une probabilité p.*

$$\lim_{n \to \infty} \frac{\boldsymbol{Eu}(\ \mathbb{E}(\Sigma_{Redist}^{Match}(L_n)))}{n} \leq \frac{p}{2}$$

Démonstration : On a

$$\mathbb{E}(\Sigma^{Match}_{Redist}(L_n)) = \frac{np}{2} + \frac{\mathbb{E}(b_1)}{2}$$

On majore le nombre de petite boîte singleton par un on a :

$$\mathbb{E}(b1) \leq \mathbb{E}(\tilde{b}1) + 1$$

Par définition on a

$$\mathbb{E}(\tilde{b}1) = \sum_S \tilde{b}1(S)\mathbb{P}(S)$$

Or d'après Karp [52] $\mathbf{Eu}(\tilde{b}1(S)) = 0(\sqrt{|S|})$ quand les objets sont i.i.d suivant une loi uniforme sur $[0,1]$.

$$\mathbf{Eu}(\ \mathbb{E}(\tilde{b}1) = \sum_S \mathbf{Eu}(\tilde{b}1(S))\mathbb{P}(S) \leq c\sqrt{n}$$

∎

4.8 Stratégie de Ré-optimisation Aléatoires

Pour l'étude de la stratégie de ré-optimisation on s'intéresse au cas où les éléments de la liste L_n sont uniformément réparties sur $[0,1]$. On introduit la variable suivante : la somme de la taille des objets $T_n = \sum_{i=1}^{n} X_i$.

On dit qu'une heuristique H_1 domine l'heuristique H_2 si la solution donnée par H_1 est au moins plus grande que celle donnée par H_2

Théorème 25 *Supposons que $X_1, ..., X_n$ sont indépendantes, identiquement distribués suivant une loi uniforme sur $[0,1]$ et que les objets sont présents avec une probabilité p.*

$$\lim_{n\to\infty} \frac{\mathbf{Eu}(\ \mathbb{E}(\Sigma^{Opt}_{Redist}))}{n} = \lim_{n\to\infty} \frac{\mathbf{Eu}(\ \mathbb{E}(\Sigma^{Match}_{Redist}(L_n)))}{n} = \frac{p}{2}$$

Démonstration : Matching est un algorithme dominant la solution optimale :

$$Opt(S) \leq Match(S) \quad \forall \quad S$$

Donc

$$\mathbb{E}(\Sigma^{Opt}_{Redist}) \leq \mathbb{E}(\Sigma^{Match}_{Redist}(L_n))$$

D'après le Théorème 24 on a $\lim_{n\to\infty} \frac{\mathbf{Eu}(\ \mathbb{E}(\Sigma^{Match}_{Redist}(L_n)))}{n} \leq \frac{p}{2}$ donc

$$\lim_{n\to\infty} \frac{\mathbf{Eu}(\ \mathbb{E}(\Sigma^{Opt}_{Redist}))}{n} \leq \frac{p}{2} \tag{4.39}$$

La solution optimale est supérieure à la taille des objets présents $T_n(S) = \sum_{i=1}^{n} X_i 1_{\{X_i \in S\}}$ (la taille des objets présents plus l'espace perdu est égale à la solution optimale) :

$$T_n(S) \leq Opt(S) \quad \forall \quad S$$

Les objets sont présents avec une probabilité p on a :

$$\sum_S \mathbb{P}(S) T_n(S) = \sum_S \mathbb{P}(S) \sum_{i=1}^{n} X_i 1_{\{X_i \in S\}} = \sum_{i=1}^{n} X_i \sum_S \mathbb{P}(S) 1_{\{X_i \in S\}} = p \sum_{i=1}^{n} X_i$$

Les objets sont i.i.d suivant une loi uniforme on a :

$$p \frac{\mathbf{Eu}(\sum_{i=1}^{n} X_i)}{n} = \frac{p}{2} \leq \lim_{n \to \infty} \frac{\mathbf{Eu}(\ \mathbb{E}(\Sigma_{Redist}^{Opt}))}{n}. \tag{4.40}$$

∎

Le résultat d'analyse en moyenne ci dessus se généralise pour les stratégies de redistributions suivant FFD et BFD d'après le lemme de karp [52].

Lemme 9 *(Karp 1982)*
Pour tout ensemble d'objet dans $[0,1]$, *l'heuristique Matching domine les heuristiques BFD et FFD.*

Corollaire 9 *Supposons que* $X_1, ..., X_n$ *sont indépendantes, identiquement distribués suivant une loi uniforme sur* $[0,1]$ *et que les objets sont présents avec une probabilité p.*

$$\lim_{n \to \infty} \frac{\boldsymbol{Eu}(\ \mathbb{E}(\Sigma_{Redist}^{FFD}(L_n)))}{n} = \lim_{n \to \infty} \frac{\boldsymbol{Eu}(\ \mathbb{E}(\Sigma_{Redist}^{BFD}(L_n)))}{n} = \frac{p}{2}$$

4.9 Conclusion

Tous les résultats établis ci dessus renforcent l'étude asymptotique pour les POCPs. La réduction de l'étude de la stratégie de distribution suivant Next-Fit-Decreasing à l'étude de somme de variable aléatoire indépendante bornée, nous a permis d'obtenir plusieurs résultats intéressant sur le comportement asymptotique de la stratégie dans le cas où $X_1, ..., X_n$ sont indépendantes, identiquement distribués suivant une loi uniforme sur $[0,1]$. L'analyse probabiliste de la stratégie de redistribution suivant Matching nous permet d'étudier le comportement asymptotique de la stratégie de ré-optimisation et d'analyser la performance en moyenne des différentes stratégies de redistribution du PBPP.

Conclusion

Dans cette conclusion, nous allons commencer par récapituler les divers questions abordées au cours de cette thèse.

Nous nous sommes intéressés par une implémentation des différentes heuristiques, pour le PVCP euclidien, afin d'obtenir des idées plus précises sur le comportement asymptotique de ce problème. Dans ce cadre, nous avons tout d'abord implémenté le recuit simulé. Après une implémentation optimale, nous avons acquis la conviction que cet algorithme n'est pas en mesure de rivaliser avec la méthode tabou. Le fait que la complexité de ces heuristiques est très grande, nous a mené à nous contenter d'une approximation de la fonctionnelle du PVCP. La comparaison de tous ces résultats laisse penser que la solution du PVC euclidien est une très bonne approximation pour le PVCP euclidien même pour des petites probabilités p.

Une grande partie de notre étude est consacrée à la recherche de cas particuliers où la nouvelle version probabiliste du problème est facile (P). Le résultat le plus général est celui obtenu pour les problèmes d'ordonnancement des travaux probabilistes à une machine : pour la première fois nous avons obtenu un problème qui reste polynomial quelque soit la probabilité. Nous avons aussi partiellement généralisé ce résultat au problème d'ordonnancement des travaux probabiliste à plusieurs machines dans le cas particulier où toutes les tâches ont la même probabilité de présence. Ainsi l'introduction des probabilités dans les POCs nous a permis d'analyser la stabilité des solutions de certains problèmes d'optimisation combinatoires quand ils sont perturbés par l'absence de certaines données.

L'étude des POTPs nous a permis de déduire une condition de stabilité de la solution du problème déterministe : les problèmes qui sont résolubles par un algorithme de liste imperturbable par l'absence de certaines données sont stables, nous remarquons qu'une telle propriété reste vraie pour les problèmes stables que nous avons proposé pour le PVCP.

Enfin nous nous sommes intéressés à la généralisation de cette méthodologie pour le Problème du Bin-Packing, ce dernier est un des rares problèmes NP-complet tel qu'on a des heuristiques très performantes. Ceci nous a encouragé à proposer des stratégies de redistribution suivant ces heuristiques. Sous la condition que les objets sont uniformément et indépendamment répartis dans $[0,1]$, nous avons pu déterminer explicitement pour la première fois le comportement en moyenne asymptotique des différentes stratégies de redistribution et de la stratégie de ré-optimisation : en analysant la stratégie de redistribu-

tion suivant "Matching", nous avons pu déduire le comportement en moyenne asymptotique de la stratégie de ré-optimisation, et des stratégies de redistribution suivant "Best-Fit-Decreasing" et "First-Fit-Decreasing". En plus pour la stratégie de redistribution suivant Next-Fit-Decreasing, nous avons obtenu plus de détail sur le comportement asymptotique : nous montrons que la fonctionnelle tend presque sûrement vers une constante (Cette constante est donnée explicitement en fonction de la constante de la fonctionnelle du problème déterministe), Une évaluation précise de la déviation de la fonctionnelle de sa moyenne est donnée, nous démontrons aussi que la fonctionnelle converge en loi.

Direction de Recherche Future

- Dans le chapitre 2 nous avons donnés des cas particuliers du PVC petit où la solution du PVC est celle du PVCP. Nous nous somme basés dans nos recherches sur des conditions pour que les deux stratégies a priori et de ré-optimisation soient équivalentes. Nous avons trouvés des conditions où ces deux stratégies sont différentes. Existe t-il des cas où la solution du PVC reste la solution du PVCP sans que les deux stratégies soient équivalentes ?

- On sait que l'algorithme de partitionnement de Karp, qu'on a décrit dans le chapitre 2, converge vers la solution optimale pour le PVC. Ce résultat reste t-il vérifié pour le PVCP ? Dans [35] on montre que la réponse dépend de la "sur-linéarité". Il serait intéressant d'aborder ce problème.

- Dans le chapitre 3, nous avons montré que la stratégie de ré-optimisation pour le POT à plusieurs machines est dans P, dans le cas de machines identiques ou uniformes. Ce résultat n'est pas encore démontré pour des machines différentes.

- Nous nous sommes limités dans le chapitre 4 à une étude asymptotique du PBPB dans le cas où la taille des objets est uniforme sur $[0, 1]$. Plusieurs généralisations sont envisageables pour ce problème. Il est intéressant d'analyser le comportement asymptotique sous d'autres lois pour la taille des objets, notamment dans le cas où la loi est uniforme sur $[a, b]$ $0 < a < b < 1$, et pour une fonction de densité symétrique...

- On sait que la classe des mesures qui donnent un packing parfait(le rapport de la moyenne de la solution optimale sur la moyenne de la taille des objets tend vers 1) est complètement caractérisée [52]. Une étude du comportement en moyenne asymptotique de la stratégie de ré-optimisation pour cette classe de mesures est souhaitable.

- Seulement des stratégies de redistribution suivant des heuristiques "off-line" sont proposées pour le PBP, l'analyse de l'algorithme "on-line", par exemple "Next-Fit" qui a l'avantage d'être linéaire, paraît possible.

- Une généralisation en dimension deux du PBPP, c'est-à-dire les objets sont des rectangles, serait intéressante.

Appendix A

Calcul de la fonctionnelle associée à la stratégie de redistribution suivant Matching

Dans cet appendice, nous donnons le calcul explicite de

$$\mathbb{P}(\tilde{x}_{\tau_j^i} \text{ seul dans une boîte } |y_{\tau_1}, ..., y_{\tau_l})$$

ainsi nous montrons que le calcul de la fonctionnelle associée à la stratégie de redistribution suivant Matching est explicite.

- On note τ_j^i le $i^{\text{ème}}$ petit objet présent entre $y_{\tau_{j-1}}$ et y_{τ_j} :

$$\tau_j^i = \sup_l \{l > \tau_j^{i-1} / y_{\tau_{j-1}} \leq \tilde{x}_l \leq y_{\tau_{j-1}} \text{et } \tilde{x}_l \text{ présent}\}$$

Lemme 10

$$\mathbb{P}(\tilde{x}_{\tau_j^i} \textit{ seul dans une boîte } |y_{\tau_1}, ..., y_{\tau_l}) = \sum_{i_j,...,i_l \in \hat{I}_j} [\prod_{k=j}^{l} \mathbb{P}(y_{\tau_k} \textit{ affecté}) \tag{A.1}$$

$$\prod_{k=j+1}^{l} \mathbb{P}(i_k \textit{ présent entre } y_{\tau_{k-1}} \textit{ et } y_{\tau_k}) \mathbb{P}(i_{l+1} \textit{ présent après } y_{\tau_l})$$

$$\sum_{k=0}^{\lfloor (N_{i,j}^l - l + j)/2 \rfloor} \binom{N_{i,j}-l+j}{2k} p^{2k} (1-p)^{N_{i,j}^l - l + j - 2k}]$$

$$\sum_{\tilde{i}_1,...,\tilde{i}_{j-1} \in \tilde{I}^j} \prod_{k=1}^{j-1} \mathbb{P}(\tilde{i}_{j-k} \textit{ présent entre } y_{\tau_{k-1}} \textit{ et } y_{\tau_k}) \mathbb{P}(\tilde{i}_0 \textit{ présent entre } y_{\tau_{j-1}} \textit{ et } \tilde{x}_{\tau_j^i})$$

où

$$\hat{I}_j = \{(i_j, ..., i_l) | i_j \geq 1, \textit{ et } i_{j+k} \geq (k+1) - (i_j + ... + i_{j-k-1}) \textit{ pour } 1 \leq k \leq l-j\}$$

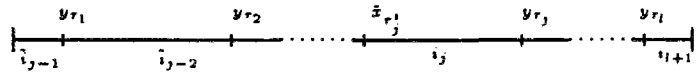
et

$$\tilde{I}^j = \{(\tilde{i}_0, \tilde{i}_1, ..., \tilde{i}_{j-1}) | \tilde{i}_0 = 0, 0 \leq \tilde{i}_{j-1} \leq j-1 \textit{ et } 0 \leq \tilde{i}_{j-k} \leq j-1-\sum_{e=1}^{k-1} \tilde{i}_{j-e} \textit{ pour } 1 \leq k \leq l-j\}$$

et

$$N_{i,j}^l = \sum_{k=j}^{l} i_k + i_{l+1}$$

Démonstration :



On remarque que $\tilde{x}_{\tau_j^i}$ est seul dans une petite boîte ssi :

- les grand objets $y_{\tau_j}, ...y_{\tau_l}$ sont tous affectés à des petits objets de taille supérieur à $\tilde{x}_{\tau_j^i}$.
- le nombre de petit objet non affectés à des grands objets, après $\tilde{x}_{\tau_j^i}$, est pair.
- les petits objets présents de taille plus petite que $\tilde{x}_{\tau_j^i}$ sont affectés.

On pose

$$D_j = \{\ y_{\tau_j}, ...y_{\tau_l} \text{ sont tous affectés }\}$$

$$E_{i,j} = \{\ \text{le nombre de petit objet non affectés après } \tilde{x}_{\tau_j^i} \text{ est pair }\}$$

$$F_{i,j} = \{\ \text{les petits objets présents de taille plus petite que } \tilde{x}_{\tau_j^i} \text{ sont affectés }\}$$

On conditionne par l' événement $D_j \cap E_{i,j}$

$$\mathbb{P}(\tilde{x}_{\tau_j^i} \text{ est dans une boîte singleton }) = \mathbb{P}(F_{i,j}|D_j \cap E_{ij})\mathbb{P}(D_j \cap E_{ij}) \qquad (A.2)$$

Calcul de $\mathbb{P}(D_j \cap E_{i,j})$:
On pose

$$D_{i,j,i_j,...i_l} = D_j \bigcap_{k=j+1}^{l} \{i_k \text{ présent entre } y_{\tau_{k-1}} \text{ et } y_{\tau_k}\} \bigcap \{i_j \text{ présent entre } x_{\tau_j^i} \text{ et } y_{\tau_j}\}$$

$$\mathbb{P}(D_i \bigcap E_{i,j}) = \sum_{i_j,...i_l} \mathbb{P}(E_{i,j}|D_{i,j,i_j,...i_l})\mathbb{P}(D_{i,j,i_j,...i_l}) \qquad (A.3)$$

$$(A.4)$$

Alors les grands objets présents après $\tilde{x}_{\tau_j^i}$ sont tous affectés ssi il y a au moins un petit objet présent de taille supérieur à $\tilde{x}_{\tau_j^i}$ non affecté, ceci se traduit par $(i_j, ...i_l) \in \hat{\mathrm{I}}_j$ où :

$$\hat{\mathrm{I}}_j = \{(i_j, ..., i_l) | i_j \geq 1, \text{ et } i_{j+k} \geq (k+1) - (i_j + ... + i_{j-k-1}) \text{ pour } 1 \leq k \leq l-j\}.$$

Les événements sont mutuellement indépendants si $(i_j, ...i_l) \in \hat{\mathrm{I}}_j$:

$$\mathbb{P}(D_{i,j,i_j,...i_l}) = \prod_{k=j}^{l} \mathbb{P}(y_{\tau_k} \text{ affecté}) \prod_{k=j+1}^{l} \mathbb{P}(i_k \text{ présent entre } y_{\tau_{k-1}} \text{ et } y_{\tau_k}) \qquad (A.5)$$

Si les grands objets présents après $\tilde{x}_{\tau_j^i}$ sont affectés alors il faut que le nombre de petit objet non affectés

après $\tilde{x}_i$ soit pair pour que $\tilde{x}_{\tau_j^i}$ soit dans une boîte singleton : on note i_{l+1} le nombre de petit objet présent après y_{τ_l}, le nombre de petit objet présent après $\tilde{x}_{\tau_j^i}$ sachant $(i_j, ..., i_l, i_{l+1})$ est

$$N_{i,j}^l = \sum_{k=j}^{l} i_k + i_{l+1}$$

le nombre de grand objets présents après $\tilde{x}_{\tau_j^i}$ est $l-j$, donc on a $l-j$ petits objets affectés à des grands objets parmi les $N_{i,j}^l$:

$$\mathbb{P}(E_{i,j}|D_{i,j,i_j,...i_l}) = \sum_{k=0}^{\lfloor (N_{i,j}-l+j)/2 \rfloor} \binom{N_{i,j}^l - l + j}{2k} p^{2k}(1-p)^{N_{i,j}-l+j-2k} \mathbb{P}(i_{l+1} \text{ petit objet présent après } y_{\tau_l}) \quad (A.6)$$

Calcul de $\mathbb{P}(F_{i,j}|D_j \cap E_{i,j})$:

On note $\tilde{i}_{j-k}$ le nombre de petits objets présents entre $y_{\tau_{k-1}}$ et y_{τ_k} ($y_{\tau_0} = 0$). On a

$$F_{i,j} = F_{i,j} \bigcap \bigcup_{\tilde{i}_1,...,\tilde{i}_{j-1}} \bigcap_{k=1}^{j-1} \{\tilde{i}_{j-k} \text{ présent entre } y_{\tau_{k-1}} \text{ et } y_{\tau_k}\} \bigcap \{\tilde{i}_0 \text{ présent entre } y_{\tau_{j-1}} \text{ et } \tilde{x}_{\tau_j^i}\}$$

On pose

$$F_{i,j,\tilde{i}_1,...,\tilde{i}_{j-1}} = F_{i,j} \bigcap \bigcap_{k=1}^{j-1} \{\tilde{i}_{j-k} \text{ présent entre } y_{\tau_{k-1}} \text{ et } y_{\tau_k}\} \bigcap \{\tilde{i}_0 \text{ présent entre } y_{\tau_{j-1}} \text{ et } \tilde{x}_{\tau_j^i}\}$$

Les événements sont deux à deux disjoints :

$$\mathbb{P}(F_{i,j}|D_j \cap E_{i,j}) = \sum_{\tilde{i}_1,...,\tilde{i}_j} \mathbb{P}(F_{i,j,\tilde{i}_1,...,\tilde{i}_{j-1}}|D_j \cap E_{i,j})$$

Alors les petits objets avant $\tilde{x}_{\tau_j^i}$ sont tous affectés sachant $\tilde{i}_0\tilde{i}_1, ..., \tilde{i}_{j-1}$ ssi $(\tilde{i}_0, \tilde{i}_1, ..., \tilde{i}_{j-1}) \in \tilde{I}^j$ où :

$$\tilde{I}^j = \{(\tilde{i}_0, \tilde{i}_1, ..., \tilde{i}_{j-1}) | \tilde{i}_0 = 0, 0 \leq \tilde{i}_{j-1} \leq j-1 \text{ et } 0 \leq \tilde{i}_{j-k} \leq j-1-\sum_{e=1}^{k-1} \tilde{i}_{j-e} \text{ pour } 1 \leq k \leq l-j\}$$

Les événements sont mutuellement indépendants d'ou :

$$\mathbb{P}(F_{i,j,\tilde{i}_0,...,\tilde{i}_{j-1}}|D_j \cap E_{i,j}) = \prod_{k=1}^{j-1} \mathbb{P}(\tilde{i}_{j-k} \text{ présent entre } y_{\tau_{k-1}} \text{ et } y_{\tau_k}) \mathbb{P}(0 \text{ présent entre } y_{\tau_{j-1}} \text{ et } \tilde{x}_{\tau_j^i}) \quad (A.7)$$

■

Bibliographie

[1] E. H. L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* Wiley, J. and Sons, New York, 1988.

[2] B. S. Baker and E. G. Coffman. A tight asymptotic bound for next-fit decreasing bin-packing. *SIAM Journal on algebraic and Discret Methods*, **2** :147–152, 1981.

[3] J. Beardwood, J. Halton, and J. Hammersley. The shortest path through many points. *Proc. Camb. Phil. Soc*, **55** :299–327, 1959.

[4] X. Berenguer. A characterisation of linear admissible transformation for the m-traveling salesmen problem. *European J.Oper.Res*, **3** :232–249, 1979.

[5] D. Bertsimas. Probabilistic combinatorial optimization problems. Technical Report **194**, Operations Research Center. MIT. Cambridge. Mass, 1988.

[6] D. Bertsimas. The probabilistic minimum spanning tree problem. *Networks*, **20** :245–275., 1990.

[7] D. Bertsimas. A vehicle routing problem with stochastic demand. *To appear in Operations Research*, 1992.

[8] D. Bertsimas and L. Howell. Further results on the probabilistic travelling salesman problem. In *MIT Sloan School of Management Working paper*, **2066-88**, September 1988.

[9] D. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Operations Research*, **38(6)** :1019–1033, 1990.

[10] E. Bonomi and J.L. Lutton. The n-city travelling salesman problem : Statistical mechanic and the metropolis algorithm. *SIAM review*, **26** :551–568, 1984.

[11] J. Bruno, E.G. Coffman, and R. Sethi. Scheduling independant tasks to reduce mean finishing time. *Communications of the ACM*, **17(7)**, 1974.

[12] J. Carlier and P. Chrétienne. Un domaine très ouvert : les problèmes d'ordonnancement. *RAIRO Recherche Operationnelle*, **16(3)** :175–217, 1982.

[13] V. Cerny. A thermodynamical approach to the travelling salesman problem : an efficient simulation algorithm. *J. Optim. Theory. Appl*, **45** :41–51, 1985.

[14] C.S. Chang, R. Nelson, and M. Pinedo. Scheduling two classes of exponential jobs on parallel processors : Structural result and worst-case analysis. *Adv.Appl.Prob*, **23** :925–944, 1991.

[15] P. Chervi. *A computational approach to probabilistic vehicle routing problems.* MIT, Master of Science in Operations Research, 1990.

[16] E. G. Coffman, G. S. Lueker, and A.H.G Rinnooy Kan. Asymptotic methods in probabilistic analysis of sequencing and packing heuristics. *Management Science*, **34(3)** :266–290, March 1988.

[17] E.G. Coffman. *Computer and Job-Shop Scheduling Theory.* Wiley, J. and Sons., 1976.

[18] R.W. Conway, W.L. Maxwel, and L.W. Miller. *Theory of scheduling.* Addison-Wesley, Reading, Mass, 1970.

[19] G. Cornuèjols and G.L. Nemhausser. Tight bounds for christofides traveling salesman heuristic. *Math . Programming*, **14** :116–121, 1978.

[20] W. L. Eastman, S. Even, and I.M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management Sci*, **11(2)** :268–279, 1964 ,Novembre.

[21] W. Feller. *An Introduction to Probability Theory and its Applications*, volume Vol. II. Wiley, J. and Sons, New York, third edition edition, 1971.

[22] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canad. J. of Math*, page 399, 1956.

[23] G.N. Frederickson. Probabilistic analysis of simple one and two-dimentional bin-packing algorithms. *Inform. Proc. Letters*, **11(4-5)** :156–161, December 1980.

[24] E.YA. Gabovitch. The small traveling salesman problem. *Trudy vychisl. Tsentra Tratu. Gos. Univ*, **19** :27–51, 1970.

[25] M. R. Garey and D.S. Johnson. *Computers and Intractability : a Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1978.

[26] K.D. Glazebrook. Scheduling tasks with exponential service times on parallel processors. *J.Appl.Prob*, **16** :685–689, 1979.

[27] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computer and Operations Research*, **13** :533–549, 1986.

[28] M. Gondran and M. Minoux. *Graphes et Algorithmes.* Collection de la direction des études et recherches d'EDF, Eyrolles, Paris, 1979.

[29] R.L. Graham. Bound for certain multiprocessing anomalies. *Bell System Technical Journal*, **45** :1563–1581, 1966.

[30] R.L. Graham. Bound for certain timing anomalies. *SIAM Journal on Applied Mathematics*, **17(2)** :416–419, 1969.

[31] M. Held and R. Karp. A dynamic programming approach to sequencing problems. *SIAM J. Appl. Math*, **10** :196–210, 1962.

[32] E. Horowitz and S. Sahni. Computing partition with applications to the knapsack problem. *Journal ACM*, **21(2)** :277–292, April 1974.

[33] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal ACM*, **23(2)** :317–327, April 1976.

[34] P. Jaillet. Probabilistic traveling salesman problem. Technical Report **185**, Operations Research Center. MIT. Cambridge. Mass., 1985.

[35] P. Jaillet. Analysis of the probabilistic traveling salesman problem in the plane. *Ricerca Operativa*, **36**, 1986.

[36] P. Jaillet. Stochastic routing problems. In G. Andreatta, F. Mason, and P. Serafini, editors, *In Stochastics in Combinatorial Optimization*, World Scientific, Singapore, 1987a.

[37] P. Jaillet. On some probabilistic combinatorial optimization problems defined on graphs. In A. Odoni, L. Bianco, and G. Szegö, editors, *Flow Control of Congested Network*, volume **38**, Berlin, 1987b. Springer Verlag.

[38] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, **36(6)** :929–936, 1988.

[39] P. Jaillet. Rates of convergence for quasi-additive smooth euclidean functionals and application to combinatorial optimization problems. *Mathematics of Operations Research*, **17 (4)** :964–980, 1992.

[40] P. Jaillet. Shortest path problems with node failures. *Networks*, **22** :589–605, 1992.

[41] P. Jaillet. Analysis of combinatorial optimization problems in the euclidean spaces. *Mathematics of Operation Research*, **18(1)**, 1993.

[42] D.S. Johnson. Fast algorithms for bin-packing. *J. Comput. system Sci*, **8** :272–314, 1974.

[43] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. optimisation by simulated annealing :an experimental evaluation ; part I graph partitioning. *Operation Research*, **37**(6) :865–892, 1989.

[44] D.S. Johnson, A. Demers, J.D. Ullman, M. R. Garey, and R.L. Graham. Worst-case performance bounds for simple one dimensional packing algorithms. *SIAM J. Comput.*, **3(4)** :298–325, 1974.

[45] T. Kampke. Optimal scheduling of jobs with exponential service times on identical parallel processors. *Operations Research*, **37** :126–133, 1989.

[46] R.M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Math. Oper. Res*, **2** :209–224, 1977.

[47] S.C. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi. Optimisation by simulated annealing. *Science*, **220** :671–680, 1983.

[48] Knödel, W. A bin-packing algorithm with complexity 0(nlogn) and performance 1 in the stochastic limit. In J. Gruska and M. Chytil, editors, *10th Symposium*, volume **118** of *Lecture Notes in computer Science*, pages 369–378, Strbske Pleso, Czechoslovakia, September 1981. Mathematical Foundations of computer Science, Springer, Berlin.

[49] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem.* John Wiley and Sons, New york, 1985.

[50] Ted. G. Lewis and H. El-Rewini. *Introduction to Parallel Computing.* Prentice-Hall,Inc, New Jersey, 1992.

[51] G.S. Lueker. An average-case analysis of bin packing with uniformly distributed item sizes. Technical Report **181**, Departement of Information and Computer Science, University of California, Irvine, 1982.

[52] G.S. Lueker and E.G. Coffman. *Probabilistic Analysis of Packing and Partitionning algorithms.* Wiley, Interscience, In discrete Mathematics and optimization, 1991.

[53] M. Lundy and A. Mees. Convergence of the annealing algorithm. *Math Prog*, **34** :111–124, 1986.

[54] M. Minoux. *Programmation Mathématique.* Dunod, Paris, 1983.

[55] R.H. Möhring, Radermacher F.J., and G. Weiss. Stochastic scheduling problems II : Set strategies. *Z.Opns.Res*, Ser A,**29** :65–104, 1980.

[56] C.A. Morgenster and H.D. Shapiro. Chromatic number approximation using simulated annealing. Technical Report **CS86-1**, Departement of Computer Science, The University of New Mexico, Albuquerque, 1986.

[57] T. W. Rhee. Probabilistic analysis of next-fit decreasing algorithm for bin-packing. *Operations Research Letters*, **6(4)** :189–191, 1987.

[58] S. Sahni. Algorithms for scheduling independent tasks. *Journal ACM*, **23(1)** :116–127, January 1976.

[59] G.H. Sasaki and B. Hajek. The time complexity of the maximum matching by simulated annealing. *J Assoc Comput.Math*, **35** :387–403, 1988.

[60] J. Steele. Subadditive euclidean functionals and nonlinear growth in geometric probability. *Annals of Probability*, **9** :365–376, 1981a.

[61] G. Weiss and M. L. Pinedo. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *J.Appl.Prob*, **17 :187–202, 1980.**